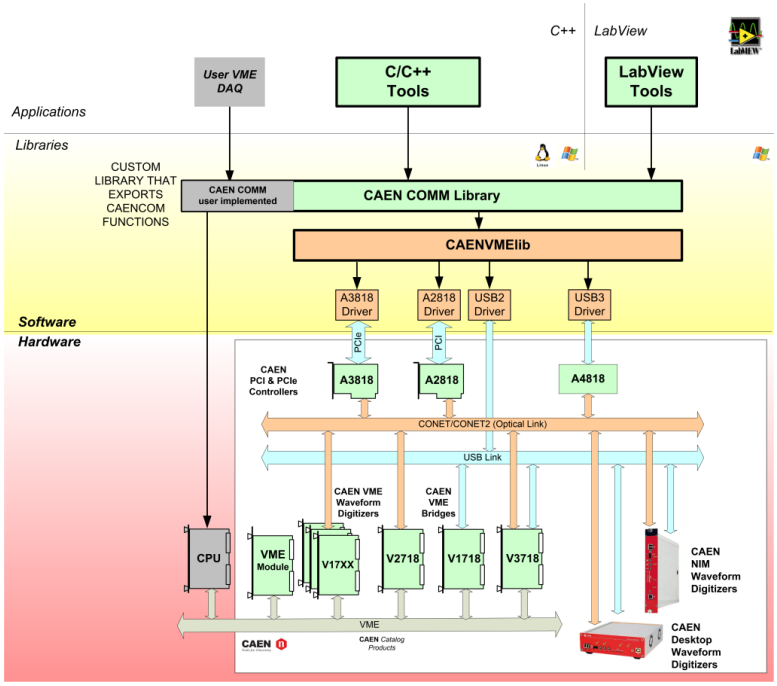
CAEN
Tools for Discovery

Electronic Instrumentation




# UM1934

# CAENComm User & Reference Manual

**Rev. 7 - April 6th, 2021**

# Purpose of this User Manual

This User's Manual contains the full description of the CAENComm library (Windows®/Linux®).

# Change Document Record

| Date | Revision | Changes |
|---|---|---|
| Previous releases of the document are not available | | |
| July 13th, 2012 | 05 | New graphical layout and added Chap. **4** |
| January 15th, 2014 | 06 | Removed "Preliminary".<br>Modified CAEN_Comm_ConnectionType removing PCI/PCIE options replaced with OpticalLink.<br>Added CAENComm_VMELIB_handle in CAENCOMM_INFO<br>Added message for developers in Sec. **System requirements & installation setup** |
| April 6th, 2021 | 07 | Added support to V3718 Bridge and A4818 Adapter. Added Chap. **5**.<br>Updated Sec. **CAENComm_ConnectionType**, Sec. **CAENComm_ErrorCode**, Sec. **CAENComm_OpenDevice** |

# Symbols, Abbreviated Terms, and Notation

| | |
|---|---|
| ADC | Analog to Digital Converter |
| DPP | Digital Pulse Processing |
| OS | Operating System |
| SBC | Single Board Computer |
| TDC | Time to Digital Converter |

# Reference Document

[RD1] AN2472 - CONET1 to CONET2 migration

[RD2] GD2512 - CAENUpgrader QuickStart Guide

[RD3] UM1935 - CAENDigitizer User and Reference Manual

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

# Index

# List of Figures



# List of Tables

# 1 CAENComm Overview

CAEN has developed a family of acquisition modules (ADC, TDC, etc.) with different standards and formats (VME, NIM and Desktop). They all provide the possibility to be handled and read out by a host PC via several communication channels. The purpose of the CAENComm library is to implement a common interface to the higher software layers, masking the details of the physical channel and its protocol, thus making the libraries and applications that rely on the CAENComm independent from the physical layer.

Moreover, the CAENComm is based on CAENVMElib, a library specifically developed for CAEN Bridges like USB-VME (Mod V1718), Optical Link-VME (Mod V2718), Optical Link/USB-VME (Mod. V3718), which implements the basic functions for accessing the VME bus (besides other specific functions for these bridges). The CAENComm library also manages the USB3-Optical Link adapter (Mod. A4818) and the PCI/PCIe Controllers (Mod. A2818/A3818).

It is then necessary that the CAENVMELib is already installed on your PC before installing the CAENComm; however, the CAENVMELib is completely transparent to the user.

CAENComm (and so the CAENDigitizer) supports the following communication channels:

- PC → USB → Digitizer series 1.0 (either Desktop or NIM models)
- PC → USB → V1718/V3718 → VME → CAEN VME Slaves (Digitizers, ADC, TDC, etc.)
- PC → USB → A4818 → CONET → Digitizer series 1.0 (all models)
- PC → PCI (A2818) → CONET → Digitizer series 1.0 (all models)
- PC → PCI (A2818) → CONET → V2718/V3718 → VME → CAEN VME Slaves (Digitizers 1.0, ADC, TDC, etc.)
- PC → PCIe (A3818) → CONET → Digitizer series 1.0 (all models)
- PC → PCIe (A3818) → CONET → V2718/V3718 → VME → CAEN VME Slaves (Digitizers 1.0, ADC, TDC, etc.)
- PC → USB → A4818 → CONET → V2718/V3718 → VME → CAEN VME Slaves (Digitizers 1.0, ADC, TDC, etc.)

**CONET** (Chainable Optical NETwork) indicates the CAEN proprietary protocol for communication on Optical Link **[RD1]**.

It is possible to develop a software for one CAEN VME card with a bus controller different from those proposed by CAEN (such as a VME-SBC); in this case it is necessary to provide a "CAENComm equivalent" library by exporting only the functions used by the software.

## System requirements & installation setup

| OS | OS version | CAEN Library required | Third-party software required |
|---|---|---|---|
| Windows | 10 | CAENVMELib | n/a |
| Linux | glibc version 2.19 or greater | | n/a |

**Tab. 1.1:** Host PC requirements

LabVIEW 2009 (only on windows and for the included LabVIEW VIs)

*Windows® is a Trademark of Microsoft Corporation in the U.S. and other countries.*

*Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.*

*LabVIEW™ is a Trademark of National Instruments Corporation.*

The hardware and software layers are reported in the scheme below.

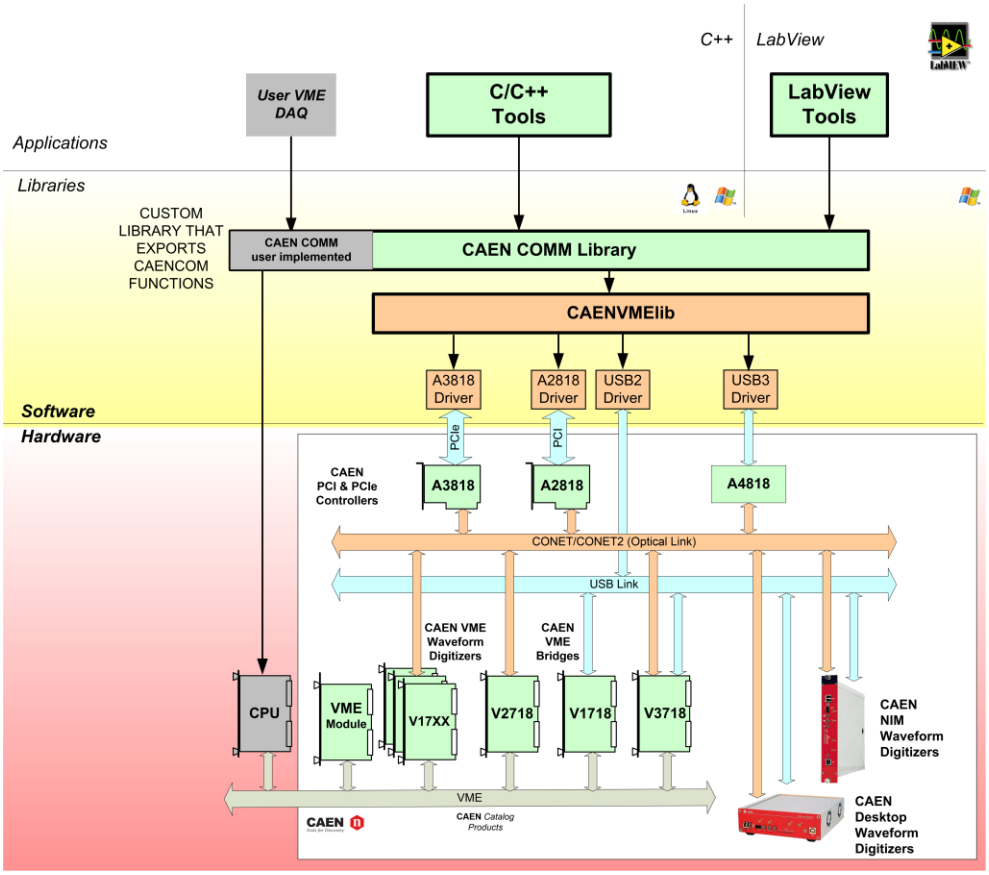



**Fig. 1.1:** Hardware and Software layers

To install the CAENComm library, follow the steps below:

- Log in to CAEN web site (www.caen.it) and download the installation package for your OS at the *CAENComm* page.
- Unpack on the host PC.
- WINDOWS USERS
  - Run the installation setup file and complete the installation wizard.
- LINUX USERS
  - Login as root.
  - Copy and unpack the needed files on your work directory.
  - Go to the library directory.
  - Execute: *sh install* (for 32-bit installation).
  - Execute: *sh install_x64* (for 64bit installation).
  - The installation copies and installs the library in */usr/lib*.

Installation instructions are also in the ReadMe file included in the library package.

**Note:** For Windows OS only, the installation of *CAENComm* also includes a demo program version in Java (*Comm/java/Demo*) and LabVIEW (*Comm/labview/Basic Example Demo*) described at Chap **4**.

**Users who developed their own software relying on a CAENComm library version less than 1.2, if they want to upgrade to 1.2 or higher, they must modify the CAENComm_ConnectionType value accordingly to the new definition!**

# 2 Function classification

CAENComm functions are divided into 4 groups:

- Device Initialization/Termination Functions
  - CAENComm_OpenDevice
  - CAENComm_CloseDevice
- Data Transfer Functions
  - CAENComm_Write32
  - CAENComm_Write16
  - CAENComm_Read32
  - CAENComm_Read16
  - CAENComm_MultiRead32
  - CAENComm_MultiRead16
  - CAENComm_MultiWrite16
  - CAENComm_MultiWrite32
  - CAENComm_BLTRead
  - CAENComm_MBLTRead
- Interrupt Handling Functions
  - CAENComm_IRQDisable
  - CAENComm_IRQEnable
  - CAENComm_IRQWait
  - CAENComm_IACKCycle
  - CAENComm_VMEIRQWait
- Information recovery functions
  - CAENComm_Info
  - CAENComm_SWRelease
  - CAENComm_DecodeError

# CAENComm Data Types

## CAENComm_ConnectionType

| Code | Value | Description |
|---|---|---|
| CAENComm_USB | 0 | Connected through USB |
| CAENComm_OpticalLink | 1 | Connected by optical link |
| CAENComm_USB_A4818_V2718 | 2 | Connectd through USB -> A4818 -> CONET -> V2718 |
| CAENComm_USB_A4818_V3718 | 3 | Connectd through USB -> A4818 -> CONET -> V3718 |
| CAENComm_USB_A4818_V4718 | 4 | COMING SOON |
| CAENComm_USB_A4818 | 5 | Connectd through USB -> A4818 |

**Tab. 2.1:** Connection Type table

## CAENComm_ErrorCode

| Error code | Value | Description |
|---|---|---|
| CAENComm_Success | 0 | Operation completed successfully |
| CAENComm_VMEBusError | -1 | VME bus error during the cycle |
| CAENComm_CommError | -2 | Communication error |
| CAENComm_GenericError | -3 | Unspecified error |
| CAENComm_InvalidParam | -4 | Invalid parameter |
| CAENComm_InvalidLinkType | -5 | Invalid Link Type |
| CAENComm_InvalidHandler | -6 | Invalid device handler |
| CAENComm_CommTimeout | -7 | Communication Timeout |
| CAENComm_DeviceNotFound | -8 | Unable to Open the requested Device |
| CAENComm_MaxDevicesError | -9 | Maximum number of devices exceeded |
| CAENComm_DeviceAlreadyOpen | -10 | The device is already opened |
| CAENComm_NotSupported | -11 | Not supported function |
| CAENComm_UnusedBridge | -12 | There aren't boards controlled by that Bridge |
| CAENComm_Terminated | -13 | Communication terminated by the Device |
| CAENComm_UnsupportedBaseAddress | -14 | The Base Address is not supported |

**Tab. 2.2:** CAENComm error codes table

## CAENCOMM_INFO

| Code | Value | Description |
|---|---|---|
| CAENComm_PCI_Board_SN | 0 | s/n of the PCI/PCIe board |
| CAENComm_PCI_Board_FwRel | 1 | Firmware Release of the PCI/PCIe board |
| CAENComm_VME_Bridge_SN | 2 | s/n of the VME bridge |
| CAENComm_VME_Bridge_FwRel1 | 3 | Firmware Release for the VME bridge |
| CAENComm_VME_Bridge_FwRel2 | 4 | Firmware Release for the optical chipset inside the VME bridge (V2718 only) |
| CAENComm_VMELIB_handle | 5 | - |

**Tab. 2.3:** CAENComm info table

## IRQ Levels

| Error code | Value | Description |
|---|---|---|
| IRQ1 | 0x01 | Interrupt level 1 |
| IRQ2 | 0x02 | Interrupt level 2 |
| IRQ3 | 0x04 | Interrupt level 3 |
| IRQ4 | 0x08 | Interrupt level 4 |
| IRQ5 | 0x10 | Interrupt level 5 |
| IRQ6 | 0x20 | Interrupt level 6 |
| IRQ7 | 0x40 | Interrupt level 7 |

**Tab. 2.4:** IRQ levels table

# 3 Function descriptions

## Initialization/termination functions

These functions allow to open and close the connection with a remote board.

To open one board is necessary to describe the 'logical' path from the PC to the device to access (one of the paths indicated in the introduction). This path is specified by the input parameters of the OpenDevice function. Once the device is opened, the function returns a handle that becomes the unique identifier of that device; any access operation to the device (except for VME cards IRQ management) will take place according to its handle, thus making transparent the physical channel.

### CAENComm_OpenDevice

**Description**
This function allows to open the device.

**Synopsis**

```
CAENComm_ErrorCode STDCALL CAENComm_CloseDevice (
                                        CAENComm_ConnectionType  LinkType,
                                        int LinkNum,
                                        int ConetNode,
                                        uint32_t VMEBaseAddress,
                                        int *handle
                                        );
```

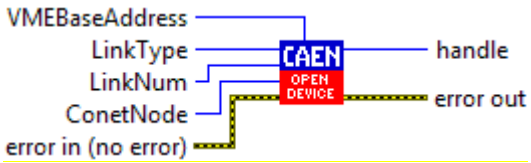
**Arguments**

| Name | Description |
|---|---|
| **LinkType** | Indicates the link used by the device:<br>0: CAENComm_USB<br>1: CAENComm_OpticalLink |
| **LinkNum** | When using OpticalLink (V2718/V3718), it is the optical link number to be used.<br>When using USB (V1718/V3718), it is the USB device number to be used.<br>When using A4818, it is the PID of the A4818 adapter. |
| **ConetNode** | For OpticalLink, it identifies which device in the daisy-chain is addressed.<br>For USB, it must be 0. |
| **VMEBaseAddress** | The VME base address of the board in case you want to access a VME device, 0 otherwise. |
| ***handle** | Pointer to the handler returned by the open function, to be used for accessing the device |

**Return Values**
0: Success; Negative numbers are error codes (see **Tab. 2.2**).

**LabVIEW Representation**

CAENComm_OpenDevice.vi



## CAENComm_CloseDevice

**Description**

This function allows to close the device

**Synopsis**

```
CAENComm_ErrorCode STDCALL CAENComm_CloseDevice(
                                        int handle
                                        );
```

**Arguments**

| Name | Description |
|---|---|
| **handle** | The handler to use for accessing the device |

**Return Values**

0: Success; Negative numbers are error codes (see **Tab. 2.2**).

**LabVIEW Representation**

CAENComm_CloseDevice.vi

# Initialization/Termination Examples

## Example N° 1

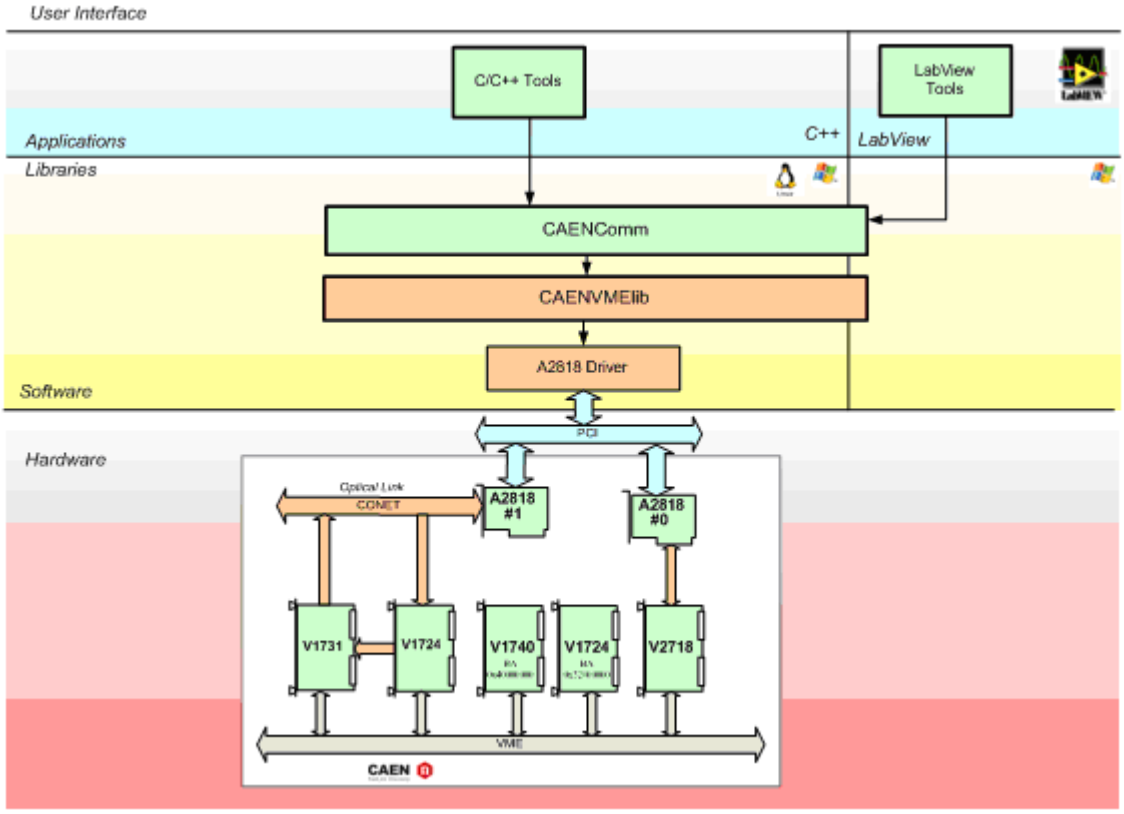



**Fig. 3.1: Block diagram of example N° 1**

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

a) Bridge CAEN V2718 physically connected via optical links to the PCI card A2818 N° 0

b) Two Digitizers CAEN (model V1724 with VME base address 0x32100000 and model V1740 with VME base address 0x40000000)

c) Two Digitizers CAEN (model V1724 and V1731 model) connected in a daisy chain between them and to the PCI card A2818 N° 1

The open for the 4 cards to access are:

**Open the V1724 (VME BASE ADDRESS 0x32100000) accessed via VMEbus through the V2718:**

```
CAENComm_OpenDevice(
Physical link       CAENComm_OpticalLink,
PCI board n.        0,
Device in chain     0,
VME Base address    0x32100000,
                    &handleV1724_0
                    );
```

**Open the V1740 (VME BASE ADDRESS 0x40000000) accessed via VMEbus through the V2718:**

```
CAENComm_OpenDevice(
Physical link       CAENComm_OpticalLink,
PCI board n.        0,
Device in chain     0,
VME Base address    0x40000000,
                    &handleV1740
                    );
```

**Open the V1724 (first in daisy chain) directly accessed via Optical Link:**

```
CAENComm_OpenDevice(
Physical link       CAENComm_OpticalLink,
PCI board n.        1,
Device in chain     0,
not used            0,
                    &handleV1724_1
                    );
```

**Open the V1731 (second in daisy chain) directly accessed via Optical Link:**

```
CAENComm_OpenDevice(
Physical link       CAENComm_OpticalLink,
PCI board n.        1,
Device in chain     1,
not used            0,
                    &handleV1731
                    );
```

## Example N° 2

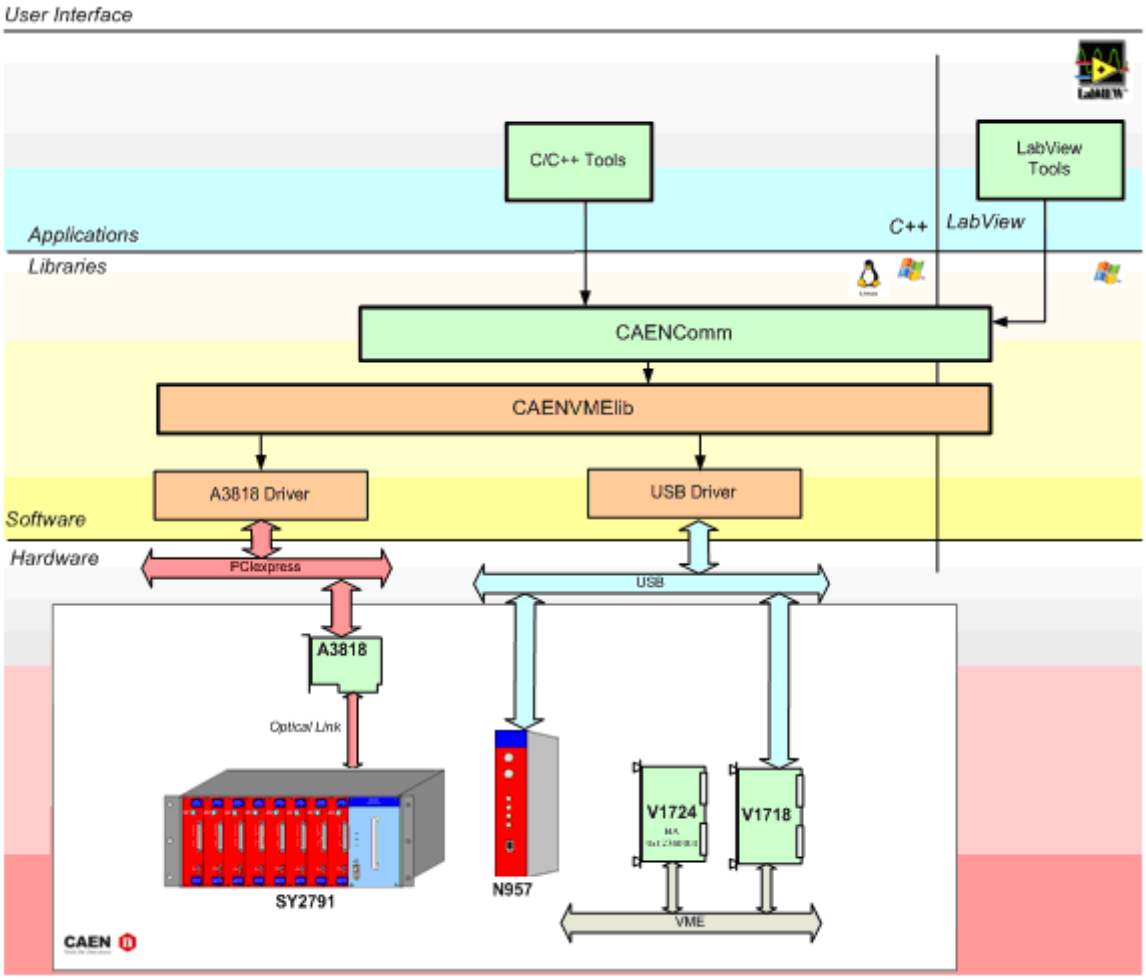



**Fig. 3.2: Block diagram of example N° 2**

Host PC houses two USB ports and a A3818 card; it is connected to three crates, respectively VME, NIM and SY2791, housing the following boards:

a) VME Crate

- Bridge CAEN V1718 physically connected to the PC via USB cable

- Digitizer model V1724 with VME base address 0x12340000

b) NIM Crate

- Multi Channel Analyzer model N957 physically connected to the PC via USB cable

c) Crate SY2791

- Model A2792 Acquisition module physically connected via optical links to the PCI Express A3818

The open for the 3 cards to access are:

**Open the V1724 accessed from VMEbus through V1718:**

```
CAENComm_OpenDevice(
Physical link       CAENComm_USB,
USB link n.         0,
not used            0,
VME Baseaddress     0x12340000,
                    &handleV1724
                    );
```

**Open the N957 connected via USB cable:**

```
CAENComm_OpenDevice(
Physical link       CAENComm_USB,
USB link n.         1,
not used            0,
not used            0,
                    &handleN957
                    );
```

**Open the SY2792 connected via Optical Link**

```
CAENComm_OpenDevice(
Physical link       CAENComm_OpticalLink,
PCIE link n.        0,
Device in chain     0,
not used            0,
                    &handleSY2792
                    );
```

# Data Transfer Functions

## CAENComm_Write32

**Description**
This function allows to write a 32 bit register of the device

**Synopsis**

```
CAENComm_ErrorCode STDCALL CAENComm_Write32(
                                    int handle,
                                    uint32_t Address,
                                    uint32_t Data
                                    );
```

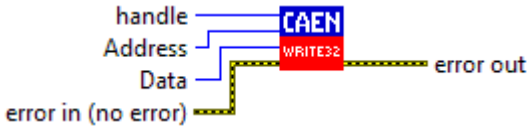
**Arguments**

| Name | Description |
|---|---|
| **handle** | Device handler |
| **Address** | Register address offset |
| **Data** | New register content to write into the device |

**Return Values**
0: Success; Negative numbers are error codes (see **Tab. 2.2**).

**LabVIEW Representation**

CAENComm_Write32.vi




## CAENComm_Write16

**Description**
This function allows to write a 16 bit register of the device.

**Synopsis**

```
CAENComm_ErrorCode STDCALL CAENComm_Write16(
                                    int handle,
                                    uint32_t Address,
                                    uint16_t Data
                                    );
```

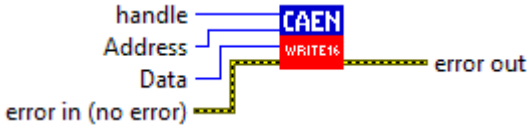
**Arguments**

| Name | Description |
|---|---|
| **handle** | Device handler |
| **Address** | Register address offset |
| **Data** | New register content to write into the device |

**Return Values**
0: Success; Negative numbers are error codes (see **Tab. 2.2**).

**LabVIEW Representation**

CAENComm_Write16.vi



## CAENComm_Read32

**Description**
This function allows to read 32 bit register of the device.

**Synopsis**
```
CAENComm_ErrorCode STDCALL CAENComm_Read32(
                                    int handle,
                                    uint32_t Address,
                                    uint32_t *Data
                                    );
```

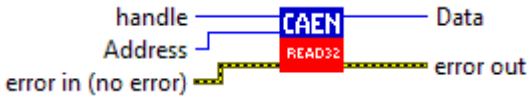
**Arguments**

| Name | Description |
|---|---|
| **handle** | Device handler |
| **Address** | Register address offset |
| **Data** | The data read from the device |

**Return Values**
0: Success; Negative numbers are error codes (see **Tab. 2.2**).

**LabVIEW Representation**

CAENComm_Read32.vi




## CAENComm_Read16

**Description**
This function allows to read 16 bit register of the device

**Synopsis**
```
CAENComm_ErrorCode STDCALL CAENComm_Read16(
                                    int handle,
                                    uint32_t Address,
                                    uint16_t *Data
                                   );
```

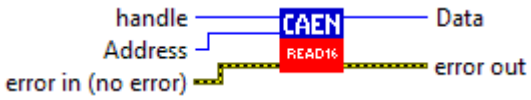
**Arguments**

| Name | Description |
|---|---|
| **handle** | Device handler |
| **Address** | Register address offset |
| **Data** | The data read from the device |

**Return Values**
0: Success; Negative numbers are error codes (see **Tab. 2.2**).

**LabVIEW Representation**

CAENComm_Read32.vi



# Multi Read/Write Functions

MultiRead and MultiWrite Functions have been developed to optimize the time in the individual accesses. In fact, both the USB in the Conet (Optical Link) foresee the exchange of request and response packets for the execution of a single read or write cycle. Because of the latency due to physical channel and protocol, the overhead of the protocol (package management) is very heavy when compared to the amount of data transferred (a 16 or 32-bit word), thus making communication ineffective. This overhead is particularly onerous in the case of the USB protocol which foresees a scheduling of the communication frames that are repeated at fixed intervals of 1ms (USB 1.1) or 125µs (USB 2.0). The purpose of MultiRead and MultiWrite is to place the requests in a single packet transmission from the PC to the device and then receive back the responses in a single package, thereby reducing the impact of latency on the single access.

NOTE: MultiRead and MultiWrite foresee an implementation at physical channel level. If a VME CAEN controller is not used, these libraries must be exported through a loop at software level running a series of individual accesses.

## CAENComm_MultiRead32

**Description**
The function performs a sequence of single 32bit Read operation.

**Synopsis**

```
CAENComm_ErrorCode STDCALL CAENComm_MultiRead32(
                                        int handle,
                                        uint32_t *Address,
                                        int nCycles,
                                        uint32_t *data,
                                        CAENComm_ErrorCode *ErrorCode
                                        );
```

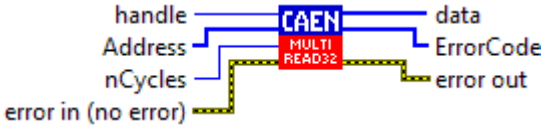
**Arguments**

| Name | Description |
|---|---|
| **handle** | Device handler |
| **Address** | Degister address offsets |
| **nCycle** | The number of Read to perform |
| **data** | The data read from the device |
| **ErrorCode** | The error codes relative to each cycle |

**Return Values**
0: Success; Negative numbers are error codes (see **Tab. 2.2**).

**LabView Representation**

CAENComm_MultiRead32.vi



## CAENComm_MultiRead16

**Description**
The function performs a sequence of single 16 bit Read operation.

**Synopsis**

```
CAENComm_ErrorCode STDCALL CAENComm_MultiRead16(
                                        int handle,
                                        uint32_t *Address,
                                        int nCycles,
                                        uint16_t *data,
                                        CAENComm_ErrorCode *ErrorCode
                                        );
```

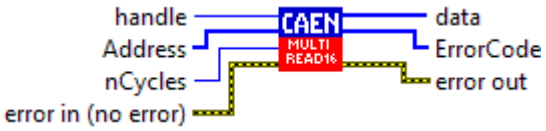
**Arguments**

| Name | Description |
|---|---|
| **handle** | Device handler |
| **Address** | Degister address offsets |
| **nCycle** | The number of Read to perform |
| **data** | The data read from the device |
| **ErrorCode** | The error codes relative to each cycle |

**Return Values**
0: Success; Negative numbers are error codes (see **Tab. 2.2**).

**LabView Representation**

CAENComm_MultiRead16.vi




## CAENComm_MultiWrite32

**Description**
The function performs a sequence of single 32 bit Write operation.

**Synopsis**

```
CAENComm ErrorCode STDCALL CAENComm MultiWrite32(
                                        int handle,
                                        uint32_t *Address,
                                        int nCycles,
                                        uint32_t *data,
                                        CAENComm_ErrorCode *ErrorCode
                                        );
```

**Arguments**

| Name | Description |
|---|---|
| **handle** | Device handler |
| **Address** | Degister address offsets |
| **nCycle** | The number of Write to perform |
| **data** | The data to write to the device |
| **ErrorCode** | The error codes relative to each cycle |

**Return Values**
0: Success; Negative numbers are error codes (see **Tab. 2.2**).

**LabView Representation**

CAENComm_MultiWrite32.vi

## CAENComm_MultiWrite16

**Description**
The function performs a sequence of single 16 bit Write operation.

**Synopsis**
```
CAENComm_ErrorCode STDCALL CAENComm_MultiWrite16(
                                        int handle,
                                        uint32_t *Address,
                                        int nCycles,
                                        uint16_t *data,
                                        CAENComm_ErrorCode *ErrorCode
                                        );
```

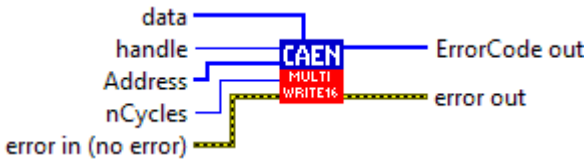
**Arguments**

| Name | Description |
|---|---|
| **handle** | Device handler |
| **Address** | Degister address offsets |
| **nCycle** | The number of Write to perform |
| **data** | The data to write to the device |
| **ErrorCode** | The error codes relative to each cycle |

**Return Values**
0: Success; Negative numbers are error codes (see **Tab. 2.2**).

**LabView Representation**

CAENComm_MultiWrite16.vi




## CAENComm_BLTRead

**Description**
This function allows to read a block of data from the device using a BLT (32 bit) cycle.

**Synopsis**
```
CAENComm_ErrorCode STDCALL CAENComm_BLTRead(
                                    int handle,
                                    uint32_t Address,
                                    uint32_t *Buff,
                                    int BltSize,
                                    int *nw
                                    );
```

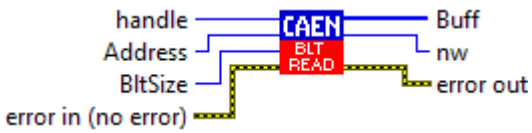
**Arguments**

| Name | Description |
|---|---|
| **handle** | Device handler |
| **Address** | Data space starting address |
| **BltSize** | Size of the Block Read Cycle (in bytes) |
| **buff** | Pointer to the read data buffer |
| **nw** | Number of longwords (32 bit) actually read from the device |

**Return Values**
0: Success; Negative numbers are error codes (see **Tab. 2.2**).

**LabView Representation**

CAENComm_BLTRead.vi



## CAENComm_MBLTRead

**Description**

This function allows to read a block of data from the device using an MBLT (64 bit) cycle.

**Synopsis**

```
CAENComm_ErrorCode STDCALL CAENComm_MBLTRead (
                                        int handle,
                                        uint32_t Address,
                                        uint32_t *Buff,
                                        int BltSize,
                                        int *nw
                                        );
```

**Arguments**

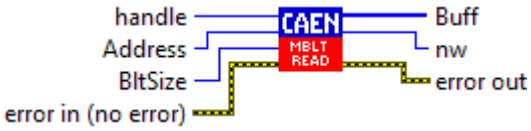
| Name | Description |
|---|---|
| **handle** | Device handler |
| **Address** | Data space starting address |
| **BltSize** | Size of the Block Read Cycle (in bytes) |
| **buff** | Pointer to the read data buffer |
| **nw** | Number of longwords (32 bit) actually read from the device |

**Return Values**

0: Success; Negative numbers are error codes (see **Tab. 2.2**).

**LabView Representation**

CAENComm_MBLTRead.vi



# Interrupt Handling Functions

## CAENComm_IRQDisable

**Description**
This function disables the IRQ lines.

**Synopsis**
```
CAENComm_ErrorCode STDCALL CAENComm_IRQDisable(
                                    int handle
                                    );
```

**Arguments**

| Name | Description |
|---|---|
| `handle` | Device handler |

**Return Values**
0: Success; Negative numbers are error codes (see **Tab. 2.2**).

**LabView Representation**

CAENComm_IRQDisable.vi


## CAENComm_IRQEnable

**Description**
This function enables the IRQ lines.

**Synopsis**
```
CAENComm_ErrorCode STDCALL CAENComm_IRQEnable(
                                    int handle
                                    );
```

**Arguments**

| Name | Description |
|---|---|
| `handle` | Device handler |

**Return Values**
0: Success; Negative numbers are error codes (see **Tab. 2.2**).

**LabView Representation**

CAENComm_IRQEnable.vi

## CAENComm_IRQWait

**Description**
The function waits the IRQ lines specified by Mask until one of them raise or timeout expires.

**Note:** This function can be used ONLY on board NOT controlled by CAEN VME Bridges.

**Synopsis**

```
CAENComm_ErrorCode STDCALL CAENComm_IRQWait(
                                    int handle,
                                    uint32_t Timeout
                                    );
```

**Arguments**

| Name | Description |
|---|---|
| **handle** | Device handler |
| **Timeout** | Timeout in milliseconds |

**Return Values**
0: Success; Negative numbers are error codes (see **Tab. 2.2**).

**LabView Representation**

CAENComm_IRQWait.vi


## CAENComm_IACKCycle

**Description**
The function performs an interrupt acknowledge cycle.

**Synopsis**

```
CAENComm_ErrorCode STDCALL CAENComm_IACKCycle(
                                    int handle,
                                    IRQLevels Level,
                                    int *BoardID
                                    );
```

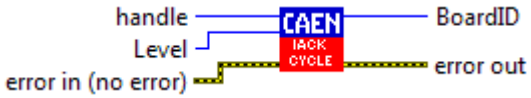
**Arguments**

| Name | Description |
|---|---|
| **handle** | Device handler |
| **Level** | The IRQ level to acknowledge |
| **BoardID** | The ID of the Board that raised the interrupt |

**Return Values**
0: Success; Negative numbers are error codes (see **Tab. 2.2**).

**LabView Representation**

CAENComm_IACKCycle.vi



## CAENComm_VMEIRQWait

**Description**
The function waits the IRQ until one of them raises or timeout expires.

**Synopsis**
```
CAENComm_ErrorCode STDCALL CAENComm_VMEIRQWait(
                                        CAENComm_ConnectionType LinkType,
                                        int LinkNum,
                                        int ConetNode,
                                        uint8_t IRQMask
                                        uint32_t Timeout,
                                        int *VMEHandle
                                        );
```

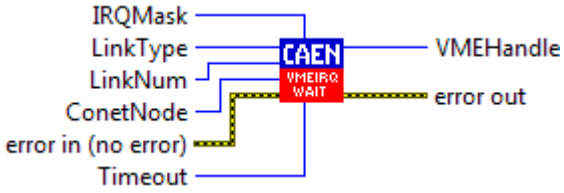
**Arguments**

| Name | Description |
|---|---|
| **LinkNum** | When using Optical Link , it is the optical link number to be used<br>When using USB, it is the USB device number to be used |
| **LinkType** | LinkType: The link used by the device<br>0: CAENComm_USB<br>1: CAENComm_OpticalLink |
| **ConetNode** | The CAEN VME Bridge number in the link |
| **IRQMask** | A bit-mask indicating the IRQ lines |
| **Timeout** | Timeout in milliseconds |
| **VMEHandle** | The CAEN Bridge handle to use in VMEIRQCheck and VMEIACKCycle |

**Return Values**
0: Success; Negative numbers are error codes (see **Tab. 2.2**).

**LabView Representation**

CAENComm_VMEIRQWait.vi




## CAENComm_VMEIRQCheck

**Description**
The function returns a bit mask indicating the active IRQ lines.

**Synopsis**
```
CAENComm_ErrorCode STDCALL CAENComm_VMEIRQCheck(
                                        int VMEhandle,
                                        uint8_t *Mask
                                        );
```

**Arguments**

| Name | Description |
|---|---|
| **VMEhandle** | CAEN Brigde handle |
| **Mask** | A bit-mask indicating the IRQ lines |

**Return Values**
0: Success; Negative numbers are error codes (see **Tab. 2.2**).

**LabView Representation**

CAENComm_VMEIRQCheck.vi

## CAENComm_VMEIACKCycle16

**Description**
The function performs a 16-bit interrupt acknowledge cycle.

**Synopsis**
```
CAENComm_ErrorCode STDCALL CAENComm_VMEIACKCycle16(
                                        int VMEhandle,
                                        IRQLevels Level,
                                        int *BoardID
                                        );
```

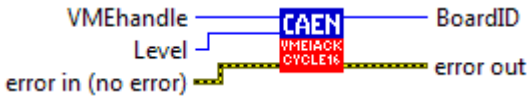
**Arguments**

| Name | Description |
|---|---|
| **VMEhandle** | CAEN Brigde handle |
| **Level** | The IRQ level to acknowledge (see IRQLevels enum) |
| **BoardID** | The Id of the Board that read the interrupt |

**Return Values**
0: Success; Negative numbers are error codes (see **Tab. 2.2**).

**LabView Representation**

CAENComm_VMEIACKCycle16.vi




## CAENComm_VMEIACKCycle32

**Description**
The function performs a 32-bit interrupt acknowledge cycle.

**Synopsis**
```
CAENComm_ErrorCode STDCALL CAENComm_VMEIACKCycle16(
                                        int VMEhandle,
                                        IRQLevels Level,
                                        int *BoardID
                                        );
```

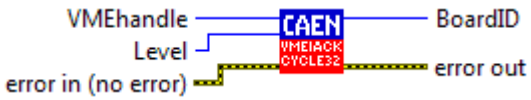
**Arguments**

| Name | Description |
|---|---|
| **VMEhandle** | CAEN Brigde handle |
| **Level** | The IRQ level to acknowledge (see IRQLevels enum) |
| **BoardID** | The Id of the Board that read the interrupt |

**Return Values**
0: Success; Negative numbers are error codes (see **Tab. 2.2**).

**LabView Representation**

CAENComm_VMEIACKCycle32.vi



# Details and Examples

The interrupts management foresees two cases:

1. The device you want to broadcast the request is directly connected to the PC.
2. The device you want to broadcast the request is accessed via bridge (only with VME).

If a device is directly connected to the optical link or USB the IRQ wait is managed by the funtion:

```
CAENComm_IRQWait(
                 int handle,
                 uint32_t Timeout
                 );
```

The function wait the IRQ until one of them raise or timeout expires.

If a device is accessed via VMEbus through the CAEN Bridge the IRQ wait is managed by the function:

```
CAENComm_VMEIRQWait(
                    int LinkType,
                    int LinkNum,
                    int ConetNode,
                    uint32_t Timeout,
                    int *VMEHandle
                    );
```

The function waits either until the bridge that manages the VME boards (specified by the function parameters) raises an IRQ or the timeout to expire.

As this function returns an VMEHandle, it is necessary to use the function CAENComm_VMEIRQCheck to acknowledge which board raised the IRQ.

EXAMPLES (for a setup like the picture shown in **Fig. 3.1**):

1) In order to handle the IRQ of boards V1740 (BA 0x40000000) and V1724 (BA 0x32100000).

IRQ manage for boards in **Fig. 3.1**:

```
CAENComm_VMEIRQWait(
                    Physical link                          CAENComm_OpticalLink,
                    PCI board n.                           0,
                    Device in chain                        0,
                    IRQmask                                0xff,
                    Timeout                                5000,
                    Handle of bridge that raises the IRQ   &VMEHandle
                    );
```

Then CAENComm_VMEIRQCheck and / or CAENComm_VMEIACKCycle32 can be used to acknowledge the activated IRQ Level.

2) In order to handle the IRQ of boards connected to the A2818#1:

```
CAENComm_IRQWait(
                 handleV1724_1,
                 Timeout  5000
                 );
```

And

```
CAENComm_IRQWait(
                 handleV1731,
                 Timeout  5000
                 );
```

# Utility Functions

## CAENComm_Info

**Description**
The function returns information about serial number or firmware release of the device.

**Synopsis**

```
CAENComm_ErrorCode STDCALL CAENComm_Info(
                                int handle,
                                CAENCOMM_INFO info,
                                char *data
                                );
```

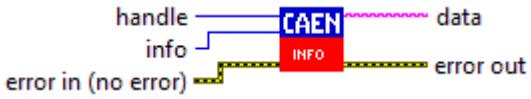
**Arguments**

| Name | Description |
|---|---|
| **handle** | Device handler |
| **info** | The interested info (see **CAENCOMM_INFO**) |
| **data** | An array (user defined to 30 byte) with the requested info |

**Return Values**
0: Success; Negative numbers are error codes (see **Tab. 2.2**).

**LabView Representation**

CAENComm_Info.vi




## CAENComm_SWRelease

**Description**
The function returns the Software Release of the library.

**Synopsis**

```
CAENComm_ErrorCode STDCALL CAENComm_SWRelease(
                                    char *SwRel
                                    );
```

**Arguments**

| Name | Description |
|---|---|
| **SwRel** | The Software Release of the library |

**Return Values**
0: Success; Negative numbers are error codes (see **Tab. 2.2**).

**LabView Representation**

CAENComm_SWRelease.vi

## CAENComm_DecodeError

**Description**

This function decodes the error code.

**Synopsis**

```
CAENComm_ErrorCode STDCALL CAENComm_DecodeError(
                                        int ErrCode,
                                        char *ErrMsg
                                        );
```

**Arguments**

| Name | Description |
|---|---|
| **ErrCode** | The error code |
| **ErrMsg** | A string with the error message |

**Return Values**

0: Success; Negative numbers are error codes (see **Tab. 2.2**).

**LabView Representation**

CAENComm_DecodeError.vi

# 4 CAENComm Demos

Once installed the CAENComm tool, two demos are available for a first approach to the library, provided both as graphical user interfaces ready to use and as source files and projects for the user development:

- Java demo (Comm/java/Demo)
- LabVIEW demo (Comm/labview/Basic Example Demo)

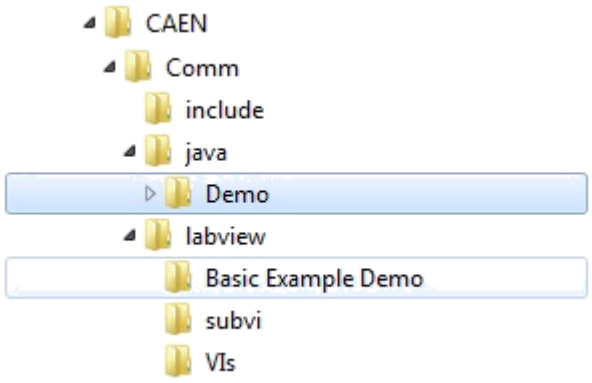



**Fig. 4.1: Folder path of the two CAENComm demos.**

The following table shows the system and software requirements needed by the demos.

| OS | OS version | CAEN Library required | Third-party software required |
|---|---|---|---|
| Windows | 10 | CAENVMELib | Java(1) Runtime Environment v6 or higher. You can download from http://www.java.com |
| | | | LabVIEW 2009(2) |

**Tab. 4.1: System and software requirements for CAENComm demo programs**

*(1) Java™ is a registered trademark of Oracle, Inc.*

*(2) LabVIEW™ is a Trademark of National Instruments Corp.*

The CAENComm demos can be used with all the CAEN digitizer series running both the standard and the DPP firmware.

# Getting Started with CAENComm Demos

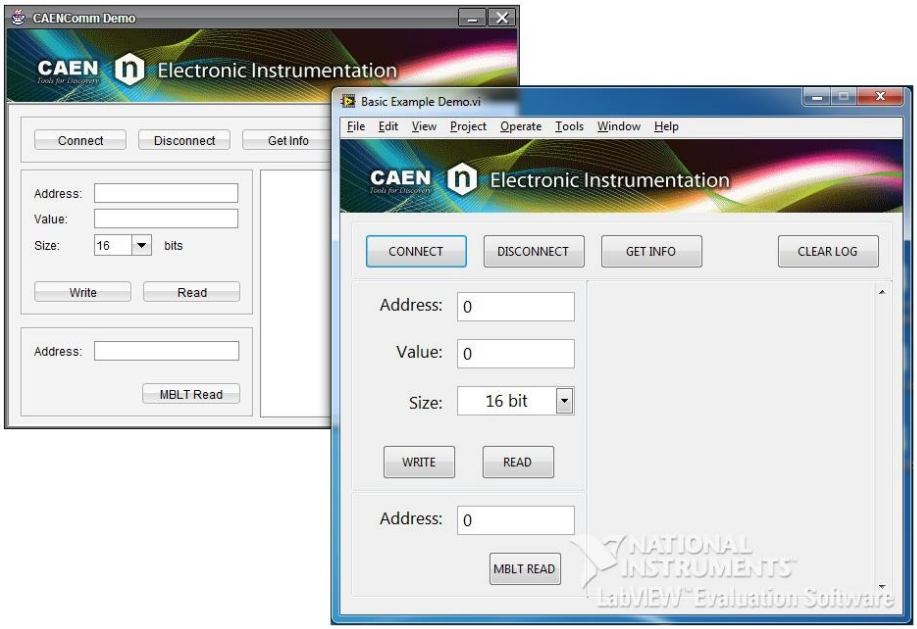



**Fig. 4.2:** CAENComm Java and LabVIEW demos

This paragraph describes how to use the functions of the CAENComm library implemented in the demos. The Java version is taken as reference; the LabVIEW version is the same.

All the following steps have been executed on a CAEN desktop digitizer DT5724 running the DPP-PHA firmware with the Optical Link (CONET2) as communication channel (through the A3818 PCIe CAEN Controller).

## Demo Structure

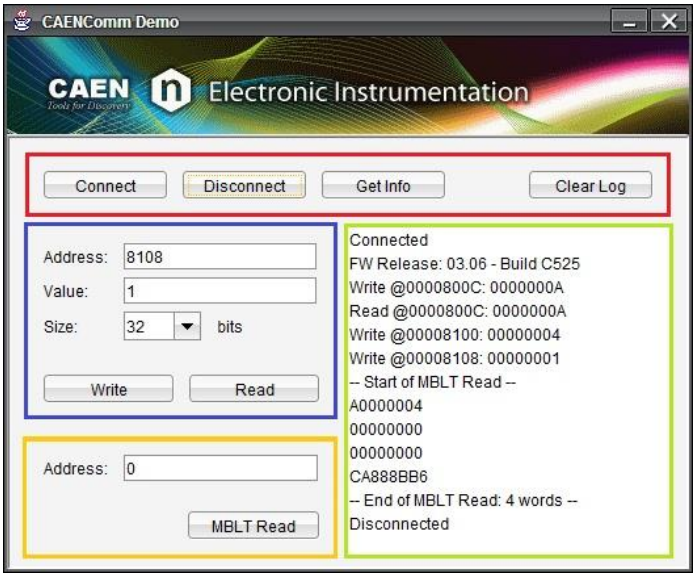



**Fig. 4.3:** CAENComm demo structure

**Control and Info buttons:** connection/disconnection handling, target FW revision information, log window clearing.

**Write/Read section:** target registers read and write mode management

**MBLT section:** target board Multi BLock Transfer read mode function

**Log window:** run time log of the actions being performed

## Launch the Demo

For the **Java** version**: launch** the **CAENCommDemo.jar** file in the CAEN/ Comm/java/Demo path.

For the **LabVIEW** version: **launch** the **Basic Example Demo.vi** file in the CAEN/ Comm/labview/Basic Example Demo path.

**Fig. 4.2** shows the GUI and VI being opened.

## Control and Info Buttons

- The **Connect** function is based on **CAENComm_OpenDevice**:

**Click** on the **CONNECT** button and **select** the proper **connection parameters** in the connection window.

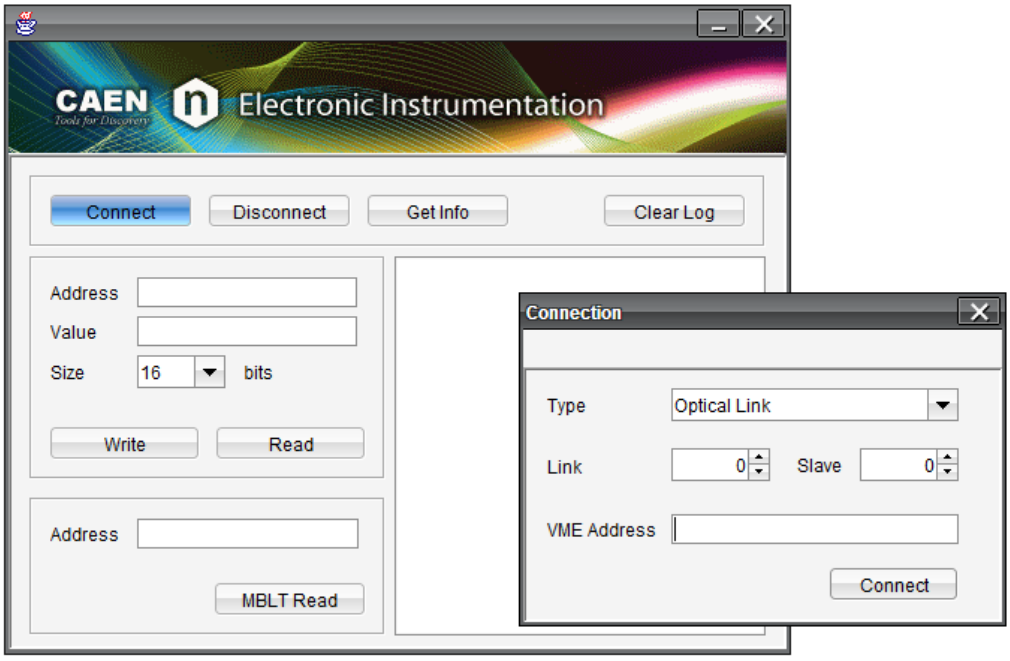



**Fig. 4.4:** Connection funtcion

**Read** in the **Log window** the result of the connection: "Connected" if succeded; an error message in case of failure (refer to **CAENComm_ErrorCode**).

Here below several connection cases and the relative settings are shown as reference.

| Connection chain | Type | Link | Slave | Address |
|---|---|---|---|---|
| PC –> **USB** –> Desktop digitizer | USB | 0 | 0 | 0 |
| PC –> **USB** –> V1718/V3718 –> **VME** –> VME digitizer | USB | 0 | 0 | VBA* |
| PC –> **PCIe** –> A3818 –> **CONET** –> V2718/V3718 –> **VME** –> VME digitizer | Optical Link | 0 | 0 | VBA* |
| PC –> **PCI** –> A2818 –> **CONET** –> NIM digitizer | Optical Link | 0 | 0 | 0 |
| PC –> **PCI** –> A2818 –> **CONET** –> VME digitizer | Optical Link | 0 | 0 | VBA* |
| PC –> **PCI** –> A2818 –> **CONET** –> VME digitizer ** | Optical Link | 0 | 1 | 0 |
| PC –> **PCIe** –> A3818 –> **CONET** –> Desktop/NIM digitizer | Optical Link | 0 | 0 | 0 |
| PC –> **USB** –> A4818 –> **CONET** –> Desktop/NIM digitizer | USB_4818 | PID**** | 0 | 0 |
| PC –> **USB** –> Desktop digitizer*** | USB | 1 | 0 | 0 |
| PC -> **USB** -> A4818 -> **CONET** - > V3718 -> VME digitizer | USB_4818_V3718 | PID**** | 0 | VBA* |
| PC -> **USB** -> A4818 -> **CONET** - > V2718 -> VME digitizer | USB_4818_V2718 | PID**** | 0 | VBA* |

**Tab. 4.2:** Examples of connection settings
* For the correct VBA (VME base address) to be used, please refer to the Digitizer documentation.
** The VME Digitizer is intended to be part of a Daisy chain (see the examples at the end of **[RD2]**).
*** It is supposed that at least two USB ports are used by the PC to communicate with as many digitizers (see the examples at the end of **[RD3]**).
**** The PID (Product Identifier) of the A4818.

- The **Get Info** function implements a single read access to the ROC FPGA register of the target board, basing on **CAENComm_Read32**:

**Click** on the **GET INFO** button to **read** the **ROC FPGA firmware release** in the Log window.


**Fig. 4.5:** Get Info function

- The **Disconnect** function is based on **CAENComm_CloseDevice**:

**Click** on the **DISCONNECT** button to **get disconnected** from the target (disconnection is confirmed by the "Disconnected" message in the Log window)


**Fig. 4.6:** Disconnect function

- The **Clear Log** is a software utility and does not base on any CAENComm function:

**Click** on the **CLEAR LOG** button to **clear** the **Log window**.

## Read/Write Target Board Registers

- The **Read** function is based on **CAENComm_Read16** and **CAENComm_Read32**, allowing a 16-bit and 32-bit single read of the target board registers.

To read the content of a register:

**Type** the **register address** (the low 16 bits of the 32-bit address) in the **ADDRESS** field (in **Fig. 4.7** the 108C address is the AMC FPGA firmware release register).

**Select** the **read access size** (32 or 16 bit) in the **SIZE** field.

**Click** on the **READ** button.

**Read** the **value of the register** in the **VALUE** field.


**Fig. 4.7:** Read function

Refer to the digitizer User Manual to decode the register information.

- The **Write** function is based on **CAENComm_Write16** and **CAENComm_Write32**, allowing a 16-bit and 32-bit single write of the target board registers.

To write a register:

**Type** the **register address** (the low 16 bits of the 32-bit address) in the **ADDRESS** field (in **Fig. 4.8** the 8100 address is the Acquisition Control register and the written value enable the acquisition run; refer the the digitizer User Manual for details).

**Type** the **value** (hexadecimal) to write in the **VALUE** field.

**Select** the **write access size** (32 or 16-bit) in the **SIZE** field.

**Click** on the **WRITE** button (a further read of the same register can be performed to check the correct writing).


**Fig. 4.8:** Write function

## Perform a MBLT Read

The MBLT Read function is based on **CAENComm_MBLTRead**, which allows to read a block of data from the targert board using an MBLT (64 bit) cycle.

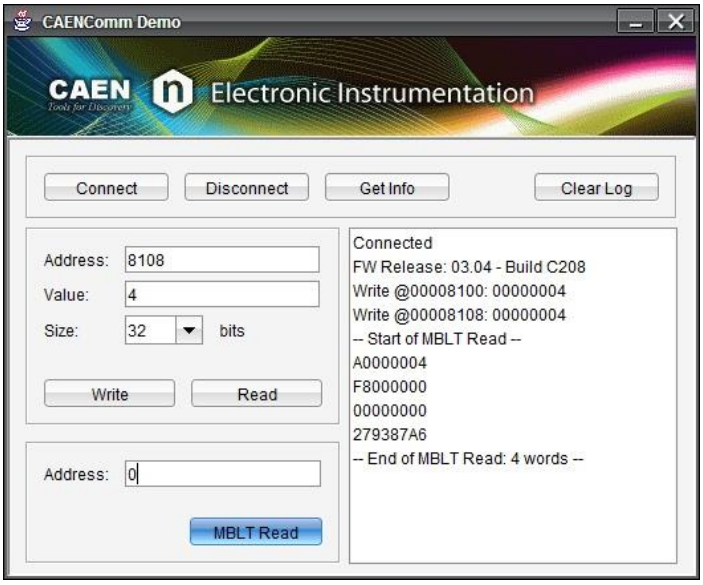
Here follows a simple example of how to perform a MBLT read:

**Use** the **Write function** to **enable** the **acquisition run** (i.e. write '4' at the adress 8100).

**Use** the **Write function** to **send** a **software trigger** to the target board (i.e. write any value at the address 8108).

**Type** the **data space starting address** in the **ADDRESS** field (default value is "**0**").

**Read data** in the **Log window**.




For details about the data format, please refer to the digitizer User Manual (if running the standard firmware) or to the DPP firmware User Manual (if running the DPP firmware).

# 5 Technical Support

CAEN makes available the technical support of its specialists for requests concerning the software and hardware. Use the support form available at the following link:

https://www.caen.it/support-services/support-form/

CAEN SpA is acknowledged as the only company in the world providing a complete range of High/Low Voltage Power Supply systems and Front-End/Data Acquisition modules which meet IEEE Standards for Nuclear and Particle Physics. Extensive Research and Development capabilities have allowed CAEN SpA to play an important, long term role in this field. Our activities have always been at the forefront of technology, thanks to years of intensive collaborations with the most important Research Centres of the world. Our products appeal to a wide range of customers including engineers, scientists and technical professionals who all trust them to help achieve their goals faster and more effectively.

**CAEN S.p.A.**
**Via Vetraia, 11**
**55049 Viareggio**
**Italy**
**Tel. +39.0584.388.398**
**Fax +39.0584.388.959**
**info@caen.it**
**www.caen.it**

**CAEN GmbH**
**Klingenstraße 108**
**D-42651 Solingen**
**Germany**
**Tel. +49 (0)212 254 4077**
**Mobile +49 (0)151 16 548 484**
**Fax +49 (0)212 25 44079**
**info@caen-de.com**
**www.caen-de.com**

**CAEN Technologies, Inc.**
**1140 Bay Street - Suite 2 C**
**Staten Island, NY 10305**
**USA**
**Tel. +1.718.981.0401**
**Fax +1.718.556.9185**
**info@caentechnologies.com**
**www.caentechnologies.com**

CAEN
Tools for Discovery
**Electronic Instrumentation**

*UM1934 - CAENComm User & Reference Manual rev. 7 - April 6th, 2021*
*00118/10:COMLib.MUTX/07*