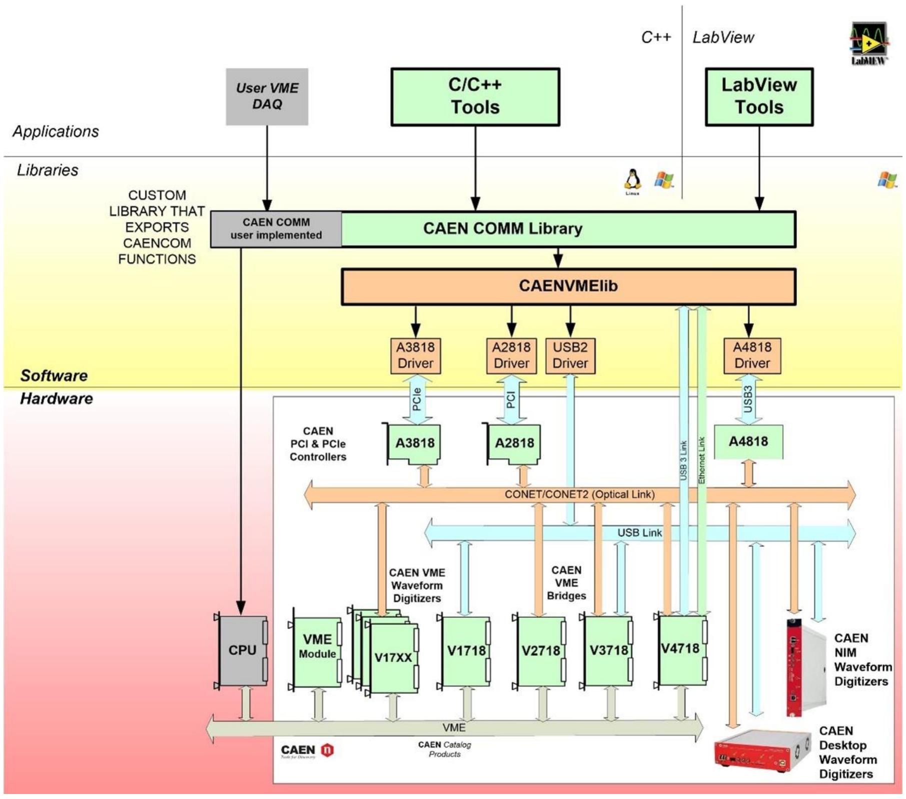
CAEN
Tools for Discovery

Electronic Instrumentation



UM7715

CAENVMELib User & Reference Manual

Rev. 2 – September 1st, 2021

Purpose of this User Manual

This User Manual contains the full description of the CAENVMELib library.

Change Document Record

Date	Revision	Changes
Sep 18th, 2020	00	First release*
May 17th, 2021	01	Added A4818 support
Sep 1st, 2021	02	Added V4718 support and CAENVME_Init2, CAENVME_DecodeError, CAENVME_DriverRelease, CAENVME_FIFOMBLTReadCycle functions

*Previous documentation on CAENVMELib used to be included in the V1718 and V2718 CAEN Bridge user manuals.

Symbols, abbreviated terms, and notation

n.a.

Reference Document

[RD1]	AN2472 – CONET1 to CONET2 migration
[RD2]	GD2512 – CAENUpgrader QuickStart Guide
[RD3]	UM7685 – V3718 User Manual
[RD4]	V1718 User Manual
[RD5]	V2718 User Manual
[RD6]	DS7799 – A4818 Data Sheet
[RD7]	UM8305 – V4718 User 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 Introduction

CAENVMELib is a set of ANSI C functions helpful for a user software development to configure and control CAEN Bridges V1718, V2718, V3718 and V4718.

All the information here described refer to CAENVMELib Rel. 3.x on, available in the following formats:

- Win32 DLL (CAEN provides the CAENVMELib.lib stub for Microsoft Visual Studio)
- Linux dynamic library

THE CAEVMELib REV. 3.1.0 OR HIGHER IS REQUIRED TO OPERATE WITH THE V3718 BRIDGE

THE CAENVMELib REV. 3.2.0 OR HIGHER IS REQUIRED TO OPERATE WITH THE A4818 ADAPTER

THE CAENVMELib REV. 3.3.0 OR HIGHER IS REQUIRED TO OPERATE WITH THE V4718 BRIDGE

CAENVMELib is logically located between an application like the samples provided and the lower layer software libraries.

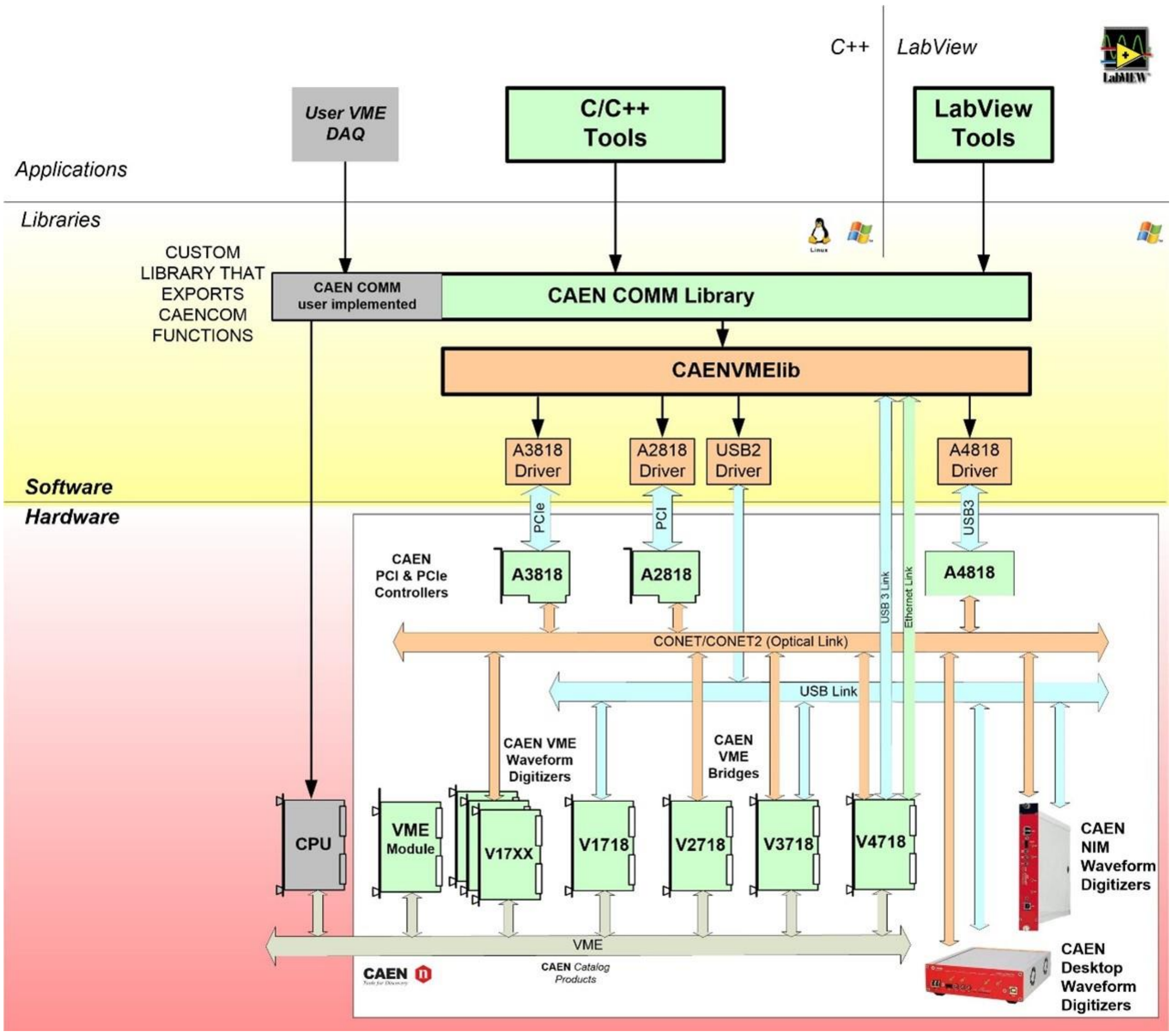


Fig. 1.1: Hardware and Software layers

2 System Requirements

Software

Compliance	CAEN SW Dependencies	Third-party software required
Windows® 8/8.1/10	CAENVMELib	No software required
Linux® glibc version 2.19 or greater		
LabVIEW™ 2009 (only for LabVIEW VIs)		NI LabVIEW Development System

Tab. 2.1: Software requirements

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.

Hardware

Communication Mode	CAEN Hardware	CAEN Driver (Windows/Linux)
USB2	V1718	V1718 USB driver
	V3718	V3718 USB driver
CONET -> VME	A2818 or A3818 and V2718, V3718 or V4718	A2818 or A3818 CONET driver
USB3 -> CONET -> VME	A4818 and V2718, V3718 or V4718	A4818 USB driver (Windows only)
USB3	V4718	No drivers required
ETHERNET	V4718	No drivers required

Tab. 2.2: Hardware requirements

3 CAENVMELib Installation

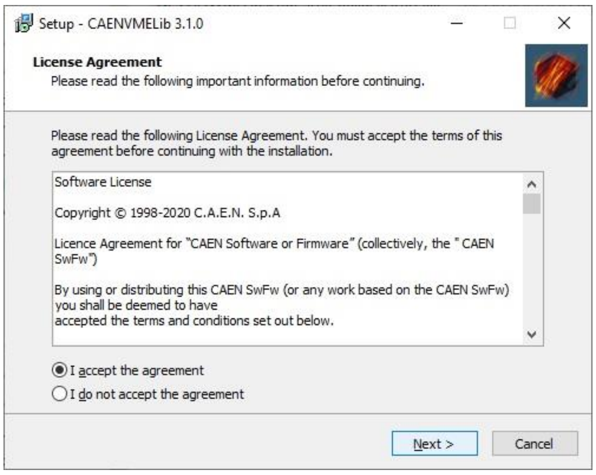
To install the CAENVMELib library, follow the steps below:

- Log in to the CAEN website (www.caen.it) and download the installation package for your OS at the *CAENVMELib* page.
- Unpack on the host PC.

Windows OS

The procedure is based on a Windows 10 64-bit system; it may be slightly different for another Windows OS.

- Run the setup file to start the Installation Wizard.
- Accept the License Agreement (**Fig. 3.1**).
- Select the Destination Location (**Fig. 3.2**).
- Select the additional component to install (**Fig. 3.3**).
- Select the Start Menu Folder (**Fig. 3.4**).
- Press the Install button to start the installation (**Fig. 3.5**).
- Complete the installation choosing to restart your computer (recommended) or not (**Fig. 3.6**).


Setup - CAENVMELib 3.1.0

License Agreement
Please read the following important information before continuing.

Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

Software License

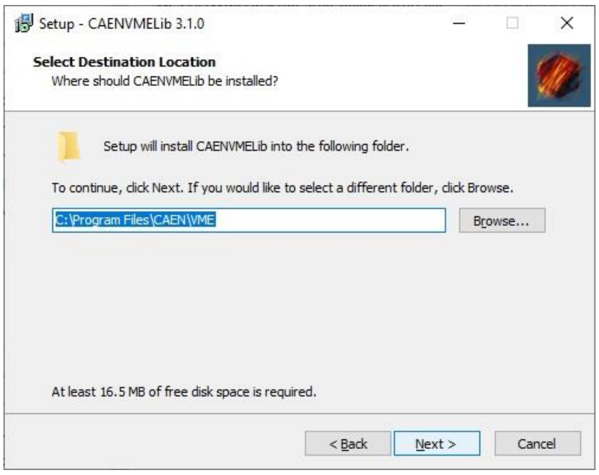
Copyright © 1998-2020 C.A.E.N. S.p.A

Licence Agreement for "CAEN Software or Firmware" (collectively, the " CAEN SwFw")

By using or distributing this CAEN SwFw (or any work based on the CAEN SwFw) you shall be deemed to have accepted the terms and conditions set out below.

[✓ I accept the agreement] I do not accept the agreement

Fig. 3.1: License Agreement step


Setup - CAENVMELib 3.1.0

Select Destination Location
Where should CAENVMELib be installed?

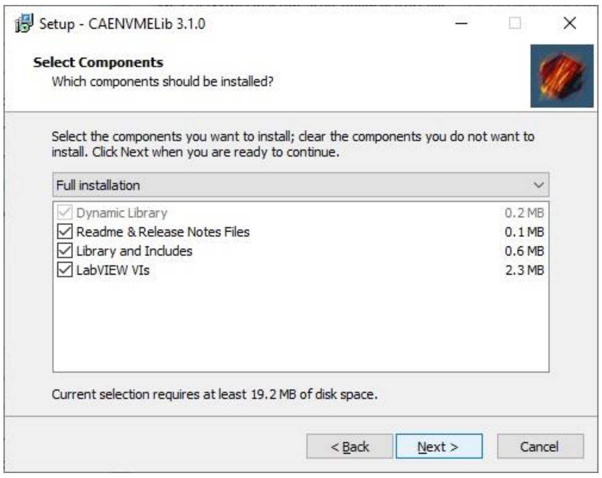
Setup will install CAENVMELib into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

C:\Program Files\CAEN\VME

At least 16.5 MB of free disk space is required.

Fig. 3.2: Select Destination Location step


Setup - CAENVMELib 3.1.0

Select Components
Which components should be installed?

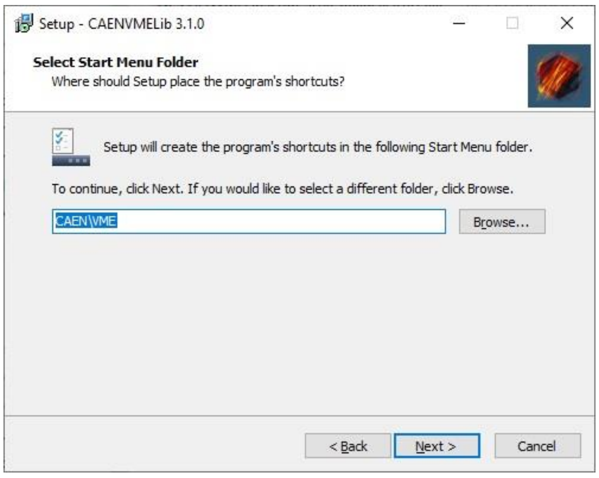
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

Full installation

✓ Dynamic Library	0.2 MB
✓ Readme & Release Notes Files	0.1 MB
✓ Library and Includes	0.6 MB
✓ LabVIEW VIs	2.3 MB

Current selection requires at least 19.2 MB of disk space.

Fig. 3.3: Select Components step


Setup - CAENVMELib 3.1.0

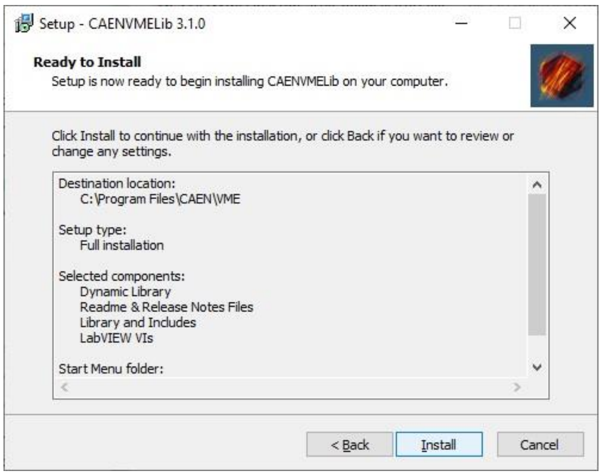
Select Start Menu Folder
Where should Setup place the program's shortcuts?

Setup will create the program's shortcuts in the following Start Menu folder.

To continue, click Next. If you would like to select a different folder, click Browse.

CAEN\VME

Fig. 3.4: Select Start Menu Folder step



Ready to Install

Setup is now ready to begin installing CAENVMELib on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

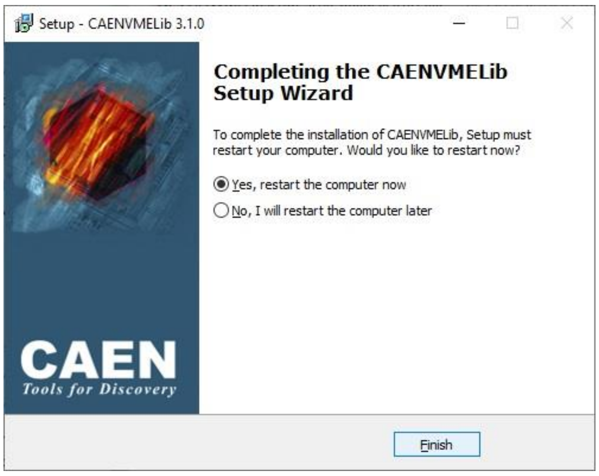
```
Destination location:
    C:\Program Files\CAEN\VME

Setup type:
    Full installation

Selected components:
    Dynamic Library
    Readme & Release Notes Files
    Library and Includes
    LabVIEW VIs

Start Menu folder:
```

Fig. 3.5: Start Installation step



Completing the CAENVMELib Setup Wizard

To complete the installation of CAENVMELib, Setup must restart your computer. Would you like to restart now?

● Yes, restart the computer now

○ No, I will restart the computer later

Fig. 3.6: Completing Installation step

Linux OS

For Linux users, the following instructions are in the README file within the library package.

- Log in as root.
- Copy the needed files on your work directory.

To install the dynamic library:

- 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*.

4 CAENVMELib Description

Return Codes

Code	Value	Description
cvSuccess	0	Operation successfully completed
cvBusError	-1	VME bus error during the cycle
cvCommError	-2	Communication error
cvGenericError	-3	Unspecified error
cvInvalidParam	-4	Invalid parameter
cvTimeoutError	-5	Timeout error
cvAlreadyOpenError	-6	The device is already open
cvMaxBoardCountError	-7	The maximum device number has been reached
cvNotSupported	8	Function not supported by that board model

Tab. 4.1: Return codes table

Functions

CAENVME_Init

Description

This function generates an opaque handle to identify a module attached to the PC. In the case of USB connection by V1718, V3718 or A4818, it must be specified only the module index (*LinkNum_or_PID*). In the case of CONET connection (by V2718 or V3718), it is required to specify also the *ConetNode* due to the possibility of an optical Daisy chain with an A2818 or A3818 controller inside the PC or through an A4818 adapter.

Synopsis

```
CAENVME_API CAENVME_Init(
                        CVBoardTypes BdType,
                        short ConetNode,
                        short LinkNum_or_PID,
                        int32_t *Handle
                        );
```

Arguments

Name	Dir.	Description
BdType	in	Indicates the model of the bridge. Values can be: – *cvV1718* (for the USB link with V1718 CAEN Bridge) – *cvV2718* (for the CONET link with V2718 CAEN Bridge) – *cvA2818* (for the CONET link with A2818 CAEN PCI Optical Controller) – *cvA2719* (for the CONET link to A2719 mezzanine of the V2718 CAEN Bridge) – *cvA3818* (for the CONET link with A3818 CAEN PCI Express Optical Controller). – *cvUSB_A4818_V2718_LOCAL* (for the CONET link to V2718 via A4818) – *cvUSB_A4818_V2718* (to link a VME slave via A4818 and V2718) – *cvUSB_A4818_LOCAL* (for the USB link to A4818 CAEN adapter) – *cvUSB_A4818_V3718_LOCAL* (for the CONET link to V3718 via A4818) – *cvUSB_A4818_V3718* (to link a VME slave via A4818 and V3718) – *cvUSB_A4818* (to link a CONET slave via A4818) – *cvUSB_A4818_A2719_LOCAL (*for the CONET link with A4818 to A2719) – *cvUSB_V3718_LOCAL* (for the USB link to V3718 CAEN bridge) – *cvPCI_A2818_V3718_LOCAL* (for the CONET link to V3718 via A2818) – *cvPCIE_A3818_V3718_LOCAL* (for the CONET link to V3718 via A3818) – *cvUSB_V3718* (to link a VME slave via USB to the V3718) – *cvPCI_A2818_V3718* (to link a VME slave via A2818 to the V3718) – *cvPCIE_A3818_V3718* (to link a VME slave via A3818 to the V3718) Refer to the CVBoardTypes enum in *CAENVMEtypes.h*
ConetNode	in	Indicates the Conet number in the daisy-chain loop (do not care in case of V1718, USB link of V3718).
LinkNum_or_PID	in	Indicates the link number, or the PID for those boards that support it (A4818/V3718)
***Handle**	out	Pointer to the handle that identifies the device.

Return Values

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

CAENVME_Init2

Description

This function generates an opaque handle to identify a module attached to the PC. It is similar to the CAENVME_Init function, but it allows to manage also the V4718 CAEN VME bridge, with the *arg* pointer allowing to manage several type of connections. In the case of CONET connection (by V2718, V3718 or V4718), it is required to specify also the *ConetNode*, due to the possibility of an optical Daisy chain with an A2818 or A3818 controller inside the PC or through an A4818 adapter.

Synopsis

```
CAENVME_API CAENVME_Init2(
                        CVBoardTypes BdType,
                        void* arg,
                        short ConetNode,
                        int32_t *Handle
                        );
```

Arguments

Name	Dir.	Description
BdType	in	Indicates the model of the bridge. Values can be: – *cvV1718* (for the USB link with V1718 CAEN Bridge) – *cvV2718* (for the CONET link with V2718 CAEN Bridge) – *cvA2818* (for the CONET link with A2818 CAEN PCI Optical Controller) – *cvA2719* (for the CONET link to A2719 mezzanine of the V2718 CAEN Bridge) – *cvA3818* (for the CONET link with A3818 CAEN PCI Express Optical Controller). – *cvUSB_A4818_V2718_LOCAL* (for the CONET link to V2718 via A4818) – *cvUSB_A4818_V2718* (to link a VME slave via A4818 and V2718) – *cvUSB_A4818_LOCAL* (for the USB link to A4818 CAEN adapter) – *cvUSB_A4818_V3718_LOCAL* (for the CONET link to V3718 via A4818) – *cvUSB_A4818_V3718* (to link a VME slave via A4818 and V3718) – *cvUSB_A4818* (to link a CONET slave via A4818) – *cvUSB_A4818_A2719_LOCAL (*for the CONET link with A4818 to A2719) – *cvUSB_V3718_LOCAL* (for the USB link to V3718 CAEN bridge) – *cvPCI_A2818_V3718_LOCAL* (for the CONET link to V3718 via A2818) – *cvPCIE_A3818_V3718_LOCAL* (for the CONET link to V3718 via A3818) – *cvUSB_V3718* (to link a VME slave via USB to the V3718) – *cvPCI_A2818_V3718* (to link a VME slave via A2818 to the V3718) – *cvPCIE_A3818_V3718* (to link a VME slave via A3818 to the V3718) – *cvUSB_V4718_LOCAL* (for the USB link to V4718 CAEN bridge) – *cvPCI_A2818_V4718_LOCAL* (for the CONET link to V4718 via A2818) – *cvPCIE_A3818_V4718_LOCAL* (for the CONET link to V4718 via A3818) – *cvETH_V4718_LOCAL* (for the Ethernet link to V4718) – *cvUSB_V4718* (to link a VME slave via USB to the V4718) – *cvPCI_A2818_V4718* (to link a VME slave via A2818 to the V4718) – *cvPCIE_A3818_V4718* (to link a VME slave via A3818 to the V4718) – *cvETH_V4718* (to link a VME slave via Ethernet to the V4718) Refer to the CVBoardTypes enum in *CAENVMEtypes.h*
arg	In	The *arg* pointer can take on different functions depending on the type of connection: – Pointer to link number, in case of an USB connection via V1718 or V3718. – Pointer to the optical link number, in case of an optical link connection via V2718, V3718 or V4718. – Pointer to the PID, in case of an USB connection to the A4818 or to the V4718. – Pointer to the IP address, in case of an Ethernet connection to the V4718
ConetNode	in	Indicates the Conet number in the daisy-chain loop (do not care in case of V1718, USB link of V3718, Ethernet and USB link of V4718).
***Handle**	out	Pointer to the handle that identifies the device.

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

CAENVME_End

Description
This function notifies the library about the end of work and frees the allocated resources.

Synopsis
```
CAENVME_API CAENVME_End(
                        int32_t Handle
                        );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.

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

CAENVME_DecodeError

Description
This function allows to decode an error code (see Sec. **Return Codes**).

Synopsis
```
CAENVME_API CAENVME_DecodeError(
                                CVErrorCodes Code
                                );
```

Arguments

Name	Dir.	Description
`Code`	in	The error code to decode.

Return Values
A string describing the error condition.

CAENVME_BoardFWRelease

Description
This function permits to read the release of the firmware loaded into the device.

Synopsis
```
CAENVME_API CAENVME_BoardFWRelease(
                                   int32_t Handle,
                                   char *FWRel
                                   );
```

Arguments

Name	Dir.	Description
`Handle`	in	The handle that identifies the device.
`FWRel`	out	Returns the firmware release of the device.

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

CAENVME_SWRelease

Description
This function permits the reading of the software release of the library.

Synopsis
```
CAENVME_API CAENVME_SWRelease(
                              char *SwRel
                              );
```

Arguments

Name	Dir.	Description
`SwRel`	out	Returns the software release of the library.

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

CAENVME_DriverRelease

Description
This function allows the reading of the software release of the device driver.

Synopsis
```
CAENVME_API CAENVME_DriverRelease(
                                  int32_t Handle,
                                  char *Rel
                                  );
```

Arguments

Name	Dir.	Description
`Handle`	in	The handle that identifies the device.
`Rel`	out	Returns the software release of the device driver.

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

CAENVME_DeviceReset

IMPLEMENTED FOR A2818, A2719, and V2718 ON LINUX PLATFORM ONLY

Description
This function permits the resetting of the device.

Synopsis
```
CAENVME_API CAENVME_DeviceReset(
                              Int_32 Handle
                              );
```

Arguments

Name	Dir.	Description
Handle	out	The handle that identifies the device.

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

CAENVME_ReadRegister

Description
This function permits to read the accessible internal registers of the Bridge.

Synopsis

```
CAENVME_API CAENVME_ReadRegister(
                                 int32_t Handle,
                                 CVRegisters Reg,
                                 unsigned int *Data
                                 );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Reg	in	The internal register to read (see CVRegisters enum in *CAENVMEtypes.h* and refer to the Bridge User Manual for a detailed registers description).
Data	out	The data read from the module.

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

CAENVME_WriteRegister

Description
This function permits to write to all accessible internal registers of the Bridge (refer to the Bridge User Manual).

Synopsis

```
CAENVME_API CAENVME_WriteRegister(
                                  int32_t Handle,
                                  CVRegisters Reg,
                                  unsigned int Data
                                  );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Reg	in	The internal register to write (see CVRegisters enum in *CAENVMEtypes.h* and refer to the Bridge User Manual for a detailed registers description).
Data	in	The data to be written to the module.

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

CAENVME_ReadCycle

Description
This function performs a single VME read cycle.

Synopsis
```
CAENVME_API CAENVME_ReadCycle(
                              int32_t Handle,
                              uint32_t Address,
                              void *Data,
                              CVAddressModifier AM,
                              CVDataWidth DW
                              );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Address	in	The VME bus address.
Data	out	The data read from the VME bus.
AM	in	The address modifier (see CVAddressModifier enum in *CAENVMEtypes.h*).
DW	in	The data width (see CVDataWidth enum in *CAENVMEtypes.h*).

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

CAENVME_WriteCycle

Description
The function performs a single VME write cycle.

Synopsis
```
CAENVME_API CAENVME_WriteCycle(
                               int32_t Handle,
                               uint32_t Address,
                               void *Data,
                               CVAddressModifier AM,
                               CVDataWidth DW
                               );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Address	in	The VME bus address.
Data	in	The data which are written to the VME bus.
AM	in	The address modifier (see CVAddressModifier enum in *CAENVMEtypes.h*).
DW	in	The data width (see CVDataWidth enum in *CAENVMEtypes.h*).

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

CAENVME_MultiRead (CONET)

Description
This function performs a sequence of VME read cycles.

Synopsis
```
CAENVME_API CAENVME_MultiRead(
                              int32_t Handle,
                              uint32_t *Addrs,
                              uint32_t *Buffer,
                              int NCycles,
                              CVAddressModifier *AMs,
                              CVDataWidth *DWs,
                              CVErrorCodes *ECs
                              );
```

Arguments

Name	Dir.	Description
handle	in	The handle that identifies the device.
Addrs	in	An array of VME bus addresses.
Buffer	out	An array of data which are read from the VME bus.
NCycles	in	The number of read cycles to perform.
AMs	in	An array of address modifiers (see CVAddressModifier enum in *CAENVMEtypes.h*).
DWs	in	An array of data widths (see CVDataWidth enum in *CAENVMEtypes.h*).
ECs	Out	The error code relative to each cycle

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

CAENVME_MultiWrite (CONET)

Description
The function performs a sequence of VME write cycles.

Synopsis
```
CAENVME_API CAENVME_ReadCycle(
                              int32_t Handle,
                              uint32_t long Addrs,
                              uint32_t *Buffer,
                              int NCycles,
                              CVAddressModifier *AMs,
                              CVDataWidth *DWs,
                              CVErrorCodes *ECs
                              );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Addrs	in	An array of VME bus addresses.
Buffer	in	An array of data written to the VME bus.
NCycles	in	The number of write cycles to perform.
AMs	in	An array of address modifiers (see CVAddressModifier enum in *CAENVMEtypes.h*).
DWs	in	An array of data widths (see CVDataWidth enum in *CAENVMEtypes.h*).
ECs	out	The error codes relative to each cycle

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

CAENVME_BLTReadCycle

Description
performs a VME block transfer read cycle. It can be used to perform MBLT transfers using 64-bit data width.

Synopsis
```
CAENVME_API CAENVME_BLTReadCycle(
                                 int32_t Handle,
                                 uint32_t Address,
                                 void *Buffer,
                                 int size,
                                 CVAddressModifier AM,
                                 CVDataWidth DW,
                                 int *count
                                 );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Address	in	The VME bus address.
Buffer	out	The data read from the VME bus.
Size	in	The size of the transfer in bytes.
AM	in	The address modifier (see CVAddressModifier enum in *CAENVMEtypes.h*).
DW	in	The data width (see CVDataWidth enum in *CAENVMEtypes.h*).
count	in	The number of bytes transferred.

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

CAENVME_RMWCycle

Description
This function performs a Read-Modify-Write cycle. The Data parameter is bidirectional: it is used to write the value to the VME bus and to return the value read.

Synopsis
```
CAENVME_API CAENVME_RMWCycle(
                             int32_t Handle,
                             uint32_t long Address,
                             void *Data,
                             CVAddressModifier AM,
                             CVDataWidth DW
                             );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Address	in	The VME bus address.
Data	in/out	The data read and then written to the VME bus.
AM	in	The address modifier (see CVAddressModifier enum in *CAENVMEtypes.h*).
DW	in	The data width (see CVDataWidth enum in *CAENVMEtypes.h*).

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

CAENVME_MBLTReadCycle

Description
The function performs a VME multiplexed block transfer read cycle.

Synopsis
```
CAENVME_API CAENVME_MBLTReadCycle(
                                  int32_t Handle,
                                  uint32_t Address,
                                  void *Buffer,
                                  int size,
                                  CVAddressModifier AM,
                                  int *count
                                  );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Address	in	The VME bus address.
Buffer	out	The data read from the VME bus.
Size	in	The size of the transfer in bytes.
AM	in	The address modifier (see CVAddressModifier enum in *CAENVMEtypes.h*).
count	out	The number of bytes transferred.

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

CAENVME_BLTWriteCycle

Description
This function performs a VME block transfer write cycle.

Synopsis
```
CAENVME_API CAENVME_BLTWriteCycle(
                                   int32_t Handle,
                                   uint32_tAddress,
                                   void *Buffer,
                                   int size,
                                   CVAddressModifier AM,
                                   CVDataWidth DW
                                   int *count
                                   );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Address	in	The VME bus address.
Buffer	in	The data to be written to the VME bus.
Size	in	The size of the transfer in bytes.
AM	in	The address modifier (see CVAddressModifier enum in *CAENVMEtypes.h*).
DW	in	The data width (see CVDataWidth enum in *CAENVMEtypes.h*).
count	out	The number of bytes transferred.

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

CAENVME_MBLTWriteCycle

Description
This function performs a VME multiplexed block transfer write cycle.

Synopsis
```
CAENVME_API CAENVME_MBLTWriteCycle(
                                   int32_t Handle,
                                   uint32_t Address,
                                   void *Buffer,
                                   int size,
                                   CVAddressModifier AM,
                                   int *count
                                   );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Address	in	The VME bus address.
Buffer	in	The data to be written to the VME bus.
Size	in	The size of the transfer in bytes.
AM	in	The address modifier (see CVAddressModifier enum in *CAENVMEtypes.h*).
count	out	The number of bytes transferred.

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

CAENVME_FIFOBLTReadCycle

Description

This function performs a VME block transfer read cycle. It can be used to perform MBLT transfers using 64-bit data width. The Address is not incremented on the VMEBus during the cycle.

Synopsis

```
CAENVME_API CAENVME_FIFOBLTReadCycle(
                                 int32_t Handle,
                                 uint32_t Address,
                                 void *Buffer,
                                 int Size,
                                 CVAddressModifier AM,
                                 CVDataWidth DW,
                                 int *count
                                 );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Address	in	The VME bus address.
Buffer	out	The data read from the VME bus.
Size	in	The size of the transfer in bytes.
AM	in	The address modifier (see CVAddressModifier enum in *CAENVMEtypes.h*).
DW	in	The data width (see CVDataWidth enum in *CAENVMEtypes.h*).
count	in	The number of bytes transferred.

Return Values

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

CAENVME_FIFOBLTWriteCycle

Description

This function performs a VME block transfer write cycle.

Synopsis

```
CAENVME_API CAENVME_FIFOBLTWriteCycle(
                                   int32_t Handle,
                                   uint32_tAddress,
                                   void *Buffer,
                                   int size,
                                   CVAddressModifier AM,
                                   CVDataWidth DW
                                   int *count
                                   );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Address	in	The VME bus address.
Buffer	in	The data to be written to the VME bus.
Size	in	The size of the transfer in bytes.
AM	in	The address modifier (see CVAddressModifier enum in *CAENVMEtypes.h*).
DW	in	The data width (see CVDataWidth enum in *CAENVMEtypes.h*).
count	out	The number of bytes transferred.

Return Values

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

CAENVME_FIFOMBLTReadCycle

Description

The function performs a VME multiplexed block transfer read cycle. The Address is not incremented on the VMEBus during the cycle.

Synopsis

```
CAENVME_API CAENVME_FIFOMBLTReadCycle(
                                 int32_t Handle,
                                 uint32_t Address,
                                 void *Buffer,
                                 int Size,
                                 CVAddressModifier AM,
                                 int *count
                                 );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Address	in	The VME bus address.
Buffer	out	The data read from the VME bus.
Size	in	The size of the transfer in bytes.
AM	in	The address modifier (see CVAddressModifier enum in *CAENVMEtypes.h*).
count	in	The number of bytes transferred.

Return Values

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

CAENVME_FIFOMBLTWriteCycle

Description

This function performs a VME multiplexed block transfer write cycle.

Synopsis

```
CAENVME_API CAENVME_FIFOMBLTWriteCycle(
                                   int32_t Handle,
                                   uint32_t Address,
                                   void *Buffer,
                                   int size,
                                   CVAddressModifier AM,
                                   int *count
                                   );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Address	in	The VME bus address.
Buffer	in	The data to be written to the VME bus.
Size	in	The size of the transfer in bytes.
AM	in	The address modifier (see CVAddressModifier enum in *CAENVMEtypes.h*).
count	out	The number of bytes transferred.

Return Values

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

CAENVME_ADOCycle

Description

This function performs a VME address only cycle. It can be used to perform MBLT transfers using 64-bit data width.

Synopsis

```
CAENVME_API CAENVME_ADOCycle(
                             int32_t Handle,
                             uint32_t Address,
                             CVAddressModifier AM
                             );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Address	in	The VME bus address.
AM	in	The address modifier (see CVAddressModifier enum in *CAENVMEtypes.h*).

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

CAENVME_ADOHCycle

Description
This function performs a VME address only with a handshake cycle.

Synopsis
```
CAENVME_API CAENVME_ADOHCycle(
                              int32_t Handle,
                              uint32_t Address,
                              CVAddressModifier AM
                              );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Address	in	The VME bus address.
AM	in	The address modifier (see CVAddressModifier enum in *CAENVMEtypes.h*).

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

CAENVME_IACKCycle

Description
This function performs a VME interrupt acknowledge cycle.

Synopsis
```
CAENVME_API CAENVME_IACKCycle(
                              int32_t Handle,
                              CVIRQLevels Level,
                              void *Vector,
                              CVDataWidth DW
                              );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Level	in	The IRQ level to acknowledge (see CVIRQLevels enum in *CAENVMEtypes.h*).
DW	in	The data width (see CVDataWidth enum *CAENVMEtypes.h*).

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

CAENVME_IRQCheck

Description
This function returns a bitmask indicating the active IRQ lines.

Synopsis
```
CAENVME_API CAENVME_IRQCheck(
                             int32_t Handle,
                             CAEN_BYTE *Mask
                             );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Mask	out	A bit-mask indicating the active IRQ lines.

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

CAENVME_IRQEnable

Description
This function enables the IRQ lines specified by a mask.

Synopsis

```
CAENVME_API CAENVME_ IRQEnble(
                              int32_t Handle,
                              int32_t Mask
                              );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Mask	in	A bit-mask indicating the IRQ lines.

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

CAENVME_IRQDisable

Description
This function disables the IRQ lines specified by Mask.

Synopsis

```
CAENVME_API CAENVME_ IRQDisable(
                                int32_t Handle,
                                int32_t Mask
                                );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Mask	in	A bit-mask indicating the IRQ lines.

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

CAENVME_IRQWait

Description
This function waits for the IRQ lines specified by the mask until one of them raises, or the timeout expires.

Synopsis

```
CAENVME_API CAENVME_IRQWait(
                            int32_t Handle,
                            uint32_t Mask,
                            uint32_t Timeout
                            );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Mask	in	A bit-mask indicating the IRQ lines.
Timeout	in	Timeout in milliseconds.

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

CAENVME_SetPulserConfig

Description
This function permits configuring the pulsers embedded on the Bridge (Pulser A and Pulser B). All the timing parameters are expressed in the specified time units.

Synopsis
```
CAENVME_API CAENVME_SetPulserConf(
                                  int32_t Handle,
                                  CVPulserSelect PulSel,
                                  unsigned char Period,
                                  unsigned char Width,
                                  CVTimeUnits Unit,
                                  unsigned char PulseNo,
                                  CVIOSources Start,
                                  CVIOSources Reset
                                  );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
PulSel	in	The pulser to configure (see CVPulserSelect enum in *CAENVMEtypes.h*).
Period	in	The period of the pulse in time units.
Width	in	The width of the pulse in time units.
Unit	in	The time unit for the pulser configuration (see CVTimeUnits enum in *CAENVMEtypes.h*).
PulseNo	in	The number of pulses to generate (0 = infinite).
Start	in	The source signal to start the pulse burst. The start signal source can optionally be front panel button or software (cvManualSW), input signal 0 (cvInputSrc0), input signal 1 (cvInputSrc1), or inputs coincidence (cvCoincidence. See CVIOSources enum in *CAENVMEtypes.h*.
Reset	in	The source signal to stop the pulse burst. The reset source signal can optionally be front panel button or software (cvManualSW) or, for pulser A the input signal 0 (cvInputSrc0), for pulser B the input signal 1 (cvInputSrc1). See CVIOSources enum in *CAENVMEtypes.h*.

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

CAENVME_StartPulser

Description
This function starts the generation of the pulse burst if the specified pulser is configured for manual/software operation (see **CAENVME_SetPulserConfig**).

Synopsis
```
CAENVME_API CAENVME_ StartPulser(
                              int32_t Handle,
                              CVPulserSelect PulSel
                              );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
PulSel	in	The pulser to configure (see CVPulserSelect enum in *CAENVMEtypes.h*).

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

CAENVME_GetPulserConf

Description
This function permits the reading of the pulsers configuration.

Synopsis

```
CAENVME_API CAENVME_GetPulserConf(
                                  int32_t Handle,
                                  CVPulserSelect PulSel,
                                  unsigned char *Period,
                                  unsigned char *Width,
                                  CVTimeUnits *Unit,
                                  unsigned char *PulseNo,
                                  CVIOSources *Start,
                                  CVIOSources *Reset
                                  );
```

Arguments

Name	Description
`Handle`	The handle that identifies the device.
`PulSel`	The pulser to configure (see CVPulserSelect enum in *CAENVMEtypes.h*).
`Period`	The period of the pulse in time units.
`Width`	The width of the pulse in time units.
`Unit`	The time unit for the pulser configuration (see CVTimeUnits enum in *CAENVMEtypes.h*).
`PulseNo`	The number of pulses to generate (0 = infinite).
`Start`	The source signal to start the pulse burst (see CVIOSources enum in *CAENVMEtypes.h*).
`Reset`	The source signal to stop the pulse burst (see CVIOSources enum in *CAENVMEtypes.h*).

Return Values

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

CAENVME_StopPulser

Description

This function stops the generation of the pulse burst if the specified pulser is configured for manual/software operation (see **CAENVME_SetPulserConfig**).

Synopsis

```
CAENVME_API CAENVME_ StopPulser(
                              int32_t Handle,
                              CVPulserSelect PulSel
                              );
```

Arguments

Name	Dir.	Description
`Handle`	in	The handle that identifies the device.
`PulSel`	in	The pulser to configure (see CVPulserSelect enum in *CAENVMEtypes.h*).

Return Values

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

CAENVME_SetScalerConf

Description

This function permits configuring the scaler embedded on the Bridge.

Synopsis

```
CAENVME_API CAENVME_SetScalerConf(
                                  int32_t Handle,
                                  short Limit,
                                  short AutoReset,
                                  CVIOSources Hit,
                                  CVIOSources Gate,
                                  CVIOSources Reset
                                  );
```

Arguments

Name	Dir.	Description
`Handle`	in	The handle that identifies the device.
`Limit`	in	The counter limit for the scaler (0 - 1024 over 10 bits).
`Autoreset`	in	Enable/disable the counter auto-reset.
`Hit`	in	The source signal for the signal to count. The hit signal source can optionally be the input signal 0 (cvInputSrc0) or input coincidence (cvCoincidence). See CVIOSources enum in *CAENVMEtypes.h*.

Gate	in	The source signal for the gate. It can optionally be front panel button or software (cvManualSW) or input signal 1 (cvInputSrc1). See CVIOSources enum in *CAENVMEtypes.h*.
Reset	in	The source signal to stop the counter. The reset signal source can optionally be the front panel button or software (cvManualSW) or input signal 1 (cvInputSrc1). See CVIOSources enum in *CAENVMEtypes.h*.

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

CAENVME_GetScalerConf

Description
This function permits the reading of the scaler configuration.

Synopsis
```
CAENVME_API CAENVME_GetScalerConf(
                                  int32_t Handle,
                                  short *Limit,
                                  short *AutoReset,
                                  CVIOSources *Hit,
                                  CVIOSources *Gate,
                                  CVIOSources *Reset
                                  );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Limit	out	The counter limit for the scaler.
AutoReset	out	The auto-reset configuration.
Hit	out	The source signal for the signal to count (see CVIOSources enum in *CAENVMEtypes.h*).
Gate	out	The source signal for the gate (see CVIOSources enum in *CAENVMEtypes.h*).
Reset	out	The source signal to stop the counter (see CVIOSources enum in *CAENVMEtypes.h*).

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

CAENVME_ResetScalerCount

Description
This function resets the counter of the scaler.

Synopsis

```
CAENVME_API CAENVME_ ResetScalerCount(
                                     int32_t Handle,
                                     );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.

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

CAENVME_EnableScalerGate

Description
This function enables the gate of the scaler.

Synopsis

```
CAENVME_API CAENVME_ EnableScalerGate(
                                      int32_t Handle,
                                      );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.

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

CAENVME_DisableScalerGate

Description
This function disables the gate of the scaler.

Synopsis

```
CAENVME_API CAENVME_ DisableScalerGate(
                                int32_t Handle,
                                );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.

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

CAENVME_SetOutputConf

Description
This function permits configuring the output lines of the Bridge. It is possible to specify the polarity for the line and the LED. The output line source depends on the line as described in **Tab. 4.2** below.

		SOURCE SELECTION			
		cdVMESignals	**cvCoincidence**	**cvMiscSignals**	**cvManualSW**
OUTPUT	0	DS	Input Coinc.	Pulser A	Manual/SW
	1	AS	Input Coinc.	Pulser A	Manual/SW
	2	DTACK	Input Coinc.	Pulser B	Manual/SW
	3	BERR	Input Coinc.	Pulser B	Manual/SW
	4	LMON	Input Coinc.	Scaler end	Manual/SW

Tab. 4.2: Source selection table for the output lines

Synopsis

```
CAENVME_API CAENVME_SetOutputConf(
                                int32_t Handle,
                                CVOutputSelect OutSel,
                                CVIOPolarity OutPol,
                                CVLEDPolarity LEDPol,
                                CVIOSources Source
                                );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
OutSel	in	The output line to configure (see CVOutputSelect enum in *CAENVMEtypes.h*).
OutPol	in	The output line polarity (see CVIOPolarity enum in *CAENVMEtypes.h*).
LEDPol	in	The output LED polarity (see CVLEDPolarity enum) in *CAENVMEtypes.h*.
Source	in	The source signal that is propagated to the output line (see CVIOSources enum in *CAENVMEtypes.h*).

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

CAENVME_GetOutputConf (CONET)

Description
This function permits the reading of the output lines configuration.

Synopsis

```
CAENVME_API CAENVME_GetOutputConf(
                                int32_t Handle,
                                CVOutputSelect OutSel,
                                CVIOPolarity *OutPol,
                                CVLEDPolarity *LEDPol,
                                CVIOSources *Source
                                );
```

Arguments

Name	Dir.	Description
`Handle`	in	The handle that identifies the device.
`OutSel`	in	The output line to configure (see CVOutputSelect enum in *CAENVMEtypes.h*).
`OutPol`	out	The output line polarity (see CVIOPolarity enum in *CAENVMEtypes.h*).
`LEDPol`	out	The output LED polarity (see CVLEDPolarity enum in *CAENVMEtypes.h*).
`Source`	out	The source signal that is propagated to the output line (see CVIOSources enum in *CAENVMEtypes.h*).

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

CAENVME_SetOutputRegister

Description
This function sets the specified lines.

Synopsis

```
CAENVME_API CAENVME_SetOutputRegister(
                                      int32_t Handle,
                                      unsigned short Mask
                                      );
```

Arguments

Name	Dir.	Description
`Handle`	in	The handle that identifies the device.
`Mask`	in	The lines to be set (refer to the CVOutputRegisterBits enum in *CAENVMEtypes.h* to compose and decode the bitmask).

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

CAENVME_ClearOutputRegister

Description
This function clears the specified lines.

Synopsis

```
CAENVME_API CAENVME_ClearOutputRegister(
                                      int32_t Handle,
                                      unsigned short Mask
                                      );
```

Arguments

Name	Dir.	Description
`Handle`	in	The handle that identifies the device.
`Mask`	in	The lines to be cleared (refer to the CVOutputRegisterBits enum in *CAENVMEtypes.h* to compose and decode the bitmask).

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

CAENVME_PulseOutputRegister

Description
This function produces a pulse with the specified lines by setting and then clearing them.

Synopsis

```
CAENVME_API CAENVME_PulseOutputRegister(
                                        int32_t Handle,
                                        unsigned short Mask
                                        );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Mask	in	The lines to be pulsed (refer to the CVOutputRegisterBits enum in *CAENVMEtypes.h* to compose and decode the bitmask).

Return Values
An error code about the execution of the function.

CAENVME_SetInputConf

Description
This function permits the configuration of the input lines of the Bridge. It is possible to specify the polarity for the line and the LED.

Synopsis
```
CAENVME_API CAENVME_SetInputConf(
                                 int32_t Handle,
                                 CVInputSelect InSel,
                                 CVIOPolarity InPol,
                                 CVLEDPolarity LEDPol
                                 );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
InSel	in	The input line to configure (see CVInputSelect enum in *CAENVMEtypes-h*).
InPol	in	The input line polarity (see CVIOPolarity enum in *CAENVMEtypes-h*).
LEDPol	in	The output LED polarity (see CVLEDPolarity enum in *CAENVMEtypes-h*).

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

CAENVME_GetInputConf

Description
This function permits the reading of the input lines configuration.

Synopsis
```
CAENVME_API CAENVME_SetInputConf(
                                 int32_t Handle,
                                 CVInputSelect InSel,
                                 CVIOPolarity *InPol,
                                 CVLEDPolarity *LEDPol
                                 );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
InSel	in	The input line to configure (see CVInputSelect enum in *CAENVMEtypes-h*).
InPol	out	The input line polarity (see CVIOPolarity enum in *CAENVMEtypes-h*).
LEDPol	out	The output LED polarity (see CVLEDPolarity enum in *CAENVMEtypes-h*).

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

CAENVME_SetArbiterType

Description
This function sets the behavior of the VME bus arbiter on the Bridge.

Synopsis

```
CAENVME_API CAENVME_SetArbiterType(
                                   int32_t Handle,
                                   CVArbiterTypes Value
                                   );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Value	in	The type of VME bus arbitration to implement (see CVArbiterTypes enum in *CAENVMEtypes.h*).

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

CAENVME_GetArbiterType

Description
This function gets the type of VME bus arbiter implemented on the Bridge.

Synopsis

```
CAENVME_API CAENVME_GetArbiterType(
                                   int32_t Handle,
                                   CVArbiterTypes *Value
                                   );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Value	out	The type of VME bus arbitration implemented (see CVArbiterTypes enum in *CAENVMEtypes.h*).

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

CAENVME_SetRequesterType

Description
This function sets the behavior of the VME bus requester on the Bridge.

Synopsis

```
CAENVME_API CAENVME_SetRequesterType(
                                     int32_t Handle,
                                     CVRequesterTypes Value
                                     );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Value	in	The type of VME bus requester to implement (see CVRequesterTypes enum in *CAENVMEtypes.h*).

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

CAENVME_GetRequesterType

Description
This function gets the type of VME bus requester implemented on the Bridge.

Synopsis

```
CAENVME_API CAENVME_GetRequesterType(
                                   int32_t Handle,
                                   CVRequesterTypes *Value
                                   );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Value	out	The type of VME bus requester implemented (see CVRequesterTypes enum in *CAENVMEtypes.h*).

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

CAENVME_SetReleaseType

Description
This function sets the release policy of the VME bus on the Bridge.

Synopsis

```
CAENVME_API CAENVME_SetReleaseType(
                                   int32_t Handle,
                                   CVReleaseTypes Value
                                   );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Value	in	The type of VME bus release policy to implement (see CVReleaseTypes enum in *CAENVMEtypes.h*).

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

CAENVME_GetReleaseType

Description
This function gets the type of VME bus release implemented on the Bridge.

Synopsis

```
CAENVME_API CAENVME_GetReleaseType(
                                   int32_t Handle,
                                   CVReleaseTypes *Value
                                   );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Value	out	The type of VME bus release policy implemented (see CVReleaseTypes enum in *CAENVMEtypes.h*).

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

CAENVME_SetBusReqLevel

Description
This function sets the specified VME bus requester priority level on the Bridge.

Synopsis

```
CAENVME_API CAENVME_SetBusReqLevel(
                                   int32_t Handle,
                                   CVBusReqLevels Value
                                   );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Value	in	The type of VME bus requester priority level to set (see CVBusReqLevels enum in *CAENVMEtypes.h*).

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

CAENVME_GetBusReqLevel

Description
This function reads the type of VME bus requester priority level implemented on the Bridge.

Synopsis

```
CAENVME_API CAENVME_GetBusReqLevel(
                                   int32_t Handle,
                                   CVBusReqLevels *Value
                                   );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Value	out	The type of VME bus requester priority level (see CVBusReqLevels enum in *CAENVMEtypes.h*).

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

CAENVME_SetTimeout

Description
This function sets the specified VME bus timeout on the Bridge.

Synopsis

```
CAENVME_API CAENVME_SetTimeout(
                               int32_t Handle,
                               CVVMETimeouts Value
                               );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Value	in	Value of VME bus timeout to set (see CVVMETimeouts enum in *CAENVMEtypes.h*).

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

CAENVME_GetTimeout

Description
This function reads the specified VME bus timeout setting of the Bridge.

Synopsis

```
CAENVME_API CAENVME_GetTimeout(
                               int32_t Handle,
                               CVVMEtimeouts *Value
                               );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Value	out	The value of VME bus timeout (see CVVMETimeouts enum in *CAENVMEtypes.h*).

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

CAENVME_SetFIFOMode

Description
This function enables/disables the auto-increment of the VME addresses during the block transfer cycles. With the FIFO mode enabled, the addresses are not incremented.

Synopsis

```
CAENVME_API CAENVME_SetFIFOMode(
                                int32_t Handle,
                                short Value
                                );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Value	in	Enable/disable the FIFO mode.

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

CAENVME_GetFIFOMode

Description
This function reads whether the auto-increment of the VME addresses during the block transfer cycles is enabled (= 0) or disabled (≠ 0).

Synopsis

```
CAENVME_API CAENVME_GetFIFOMode(
                               int32_t Handle,
                               short *Value
                               );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Value	out	The FIFO mode read setting.

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

CAENVME_ReadDisplay

Description
This function reads the VME data display on the front panel of the module.

Synopsis

```
CAENVME_API CAENVME_ReadDisplay(
                                int32_t Handle,
                                CVDisplay *Value
                                );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Value	out	The values read out from the module (see CVDisplay enum in *CAENVMEtypes.h* to decode the returned value).

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

CAENVME_SetLocationMonitor

Description
This function sets the Location Monitor.

Synopsis

```
CAENVME_API CAENVME_SetLocationMonitor(
                                       int32_t Handle,
                                       uint32_t Address,
                                       CVAddressModifier Am,
                                       short Write,
                                       short Lword,
                                       short Iack
                                       );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Address	in	The address to be monitored.
AM	in	The address modifier (see CVAddressModifier enum in *CAENVMEtypes.h*).
Write	in	Flag to specify read or write cycle types.
Lword	in	Flag to specify long-word cycle type.
Iack	in	Flag to specify interrupt acknowledge cycle type.

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

CAENVME_SystemReset

Description
This function performs a system reset on the Bridge.

Synopsis

```
CAENVME_API CAENVME_SystemReset(
                               int32_t Handle,
                                );
```

Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.

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

CAENVME_BLTReadAsync

THIS FUNCTION CANNOT BE USED WITH THE V1718, V3718 AND V4718 BRIDGES

THIS FUNCTION IS IMPLEMENTED ON LINUX PLATFORM ONLY

Description
This function starts a VME block transfer read cycle. It can be used to perform MBLT transfers using 64-bit data width. Please, take care to call the CAENVME_BLTReadWait function before any other call to a CAENVMElib function with the same handle.

Synopsis

```
CAENVME_API CAENVME_BLTReadAsync(
                                 int32_t Handle,
                                 uint32_t Address,
                                 void *Buffer,
                                 int Size,
                                 CVAddressModifier AM,
                                 CVDataWidth DW
                                 );
```

Arguments

Name	Dir.	Description
`Handle`	in	The handle that identifies the device.
`Address`	in	The VME bus address.
`Buffer`	out	The data read from the VME bus.
`Size`	in	The size of the transfer in bytes.
`AM`	in	The address modifier (see CVAddressModifier enum in *CAENVMEtypes.h*).
`DW`	in	The data width (see CVDataWidth enum in *CAENVMEtypes.h*).

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

CAENVME_BLTReadWait

THIS FUNCTION CANNOT BE USED WITH THE V1718, V3718 AND V4718 BRIDGES

THIS FUNCTION IS IMPLEMENTED ON LINUX PLATFORM ONLY

Description
This function waits for the completion of a VME block transfer read cycle started with the CAENVME_BLTReadAsync function call.

Synopsis

```
CAENVME_API CAENVME_BLTReadAsync(
                                 int32_t Handle,
                                 int *Count
                                 );
```

Arguments

Name	Dir.	Description
`Handle`	in	The handle that identifies the device.
`Count`	out	The number of bytes transferred.

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

5 CAENVME Demos

CAEN provides simple demos based on the functions of the CAENVMELib to demonstrate how to control CAEN Bridges (V1718/VX1718, V2718/VX2718, V3718/VX3718, V4718/VX4718) and giving to Users a starting point for the development of their applications. Demo versions are available in C/C++ source code (for Windows and Linux OS), LabVIEW, and .NET with friendly graphical interfaces (Windows OS only).

Users find the CAENVME demo console version included in the Linux package of the CAENVMELib library, while Windows Users find all the available versions (console, LabVIEW, and .NET graphic) in a unique package free downloadable at the “CAEN VME Demos” page once they login to CAEN web site (www.caen.it).

In the following section, the CAENVME .NET Demo is described in detail, considering the LabVIEW version is very similar, while the console version is self-explicative.

CAENVME .NET Demo (Windows only)

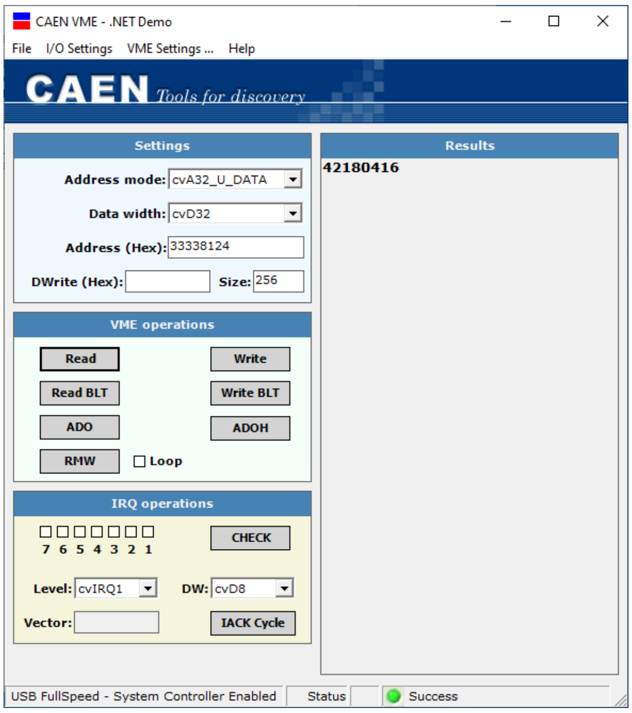
It is a C/C++ user-friendly interface for CAEN Bridges control which requires Microsoft .NET Framework 2.0 or later. The demo contains a Wrapper library that allows CAENVMELib functions to be managed by .NET applications.

- Launch the CAENVMEDemoDotNet installer file and complete the installation wizard.
- The demo can be run by the desktop shortcut or by the CAENVMEDemoDotNet executable file in the demo directory.

Main Menu

The Main Menu allows to perform and monitor the supported Data and IRQ cycles.

- *Data cycles*: once the Address Mode and the Data Width are selected, the User has to write the hexadecimal address where the cycle must be performed, the possible datum to be written (DWrite), and the Size; then, the “VME operations” buttons allow to execute the desired cycle (Read, Write, ReadBLT, WriteBLT, ADO, ADOH, RMW) that can optionally be looped (Loop). The operation results are shown in the side “Results” white area. The status bar at the bottom of the window signals possible errors on the bus.
- *IRQ cycles*: in the “IRQ operations” section, seven check cells (1 to 7) allow the detecting of an input request on the bus by checking the relevant cell; the remaining fields allow to broadcast an interrupt acknowledge CYCLE.



Address mode: cvA32_U_DATA
Data width: cvD32
Address (Hex): 33338124
DWrite (Hex):
Size: 256

42180416

Level: cvIRQ1
DW: cvD8
Vector:

USB FullSpeed - System Controller Enabled | Status | Success

Fig. 5.1: The demo Main Menu

CAEN VME Settings

The CAEN VME Settings Menu allows performing the VME general settings of V1718, V2718, V3718 and V4718 Bridges which are described in detail in the “VME Interface” chapter of the Bridge User Manual**[RD3][RD4][RD5]**.

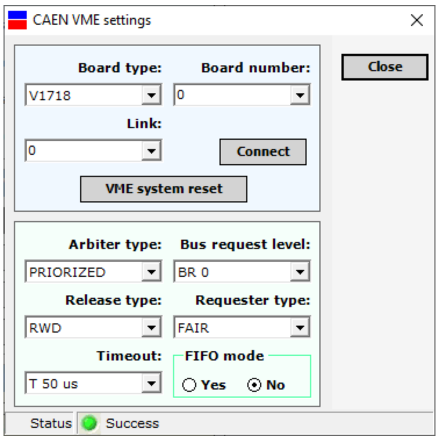
Board Type must be set to:

- “V2718”, when using the Optical Link with V2718, V3718 and V4718;
- “V1718”, when using the USB Link with V1718 and V3718.

Link is the PCI or PCI Express link number:

- “0” in case of the A2818 PCI controller;
- “0”, “1,”,”2”,”3” in the case of the A3818 PCI Express controller.

Board Number is the Conet node, that is the V2718/V3718/V4718 position in case of optical Daisy chain connection.



Board type: V1718
Board number: 0
Link: 0
Arbiter type: PRIORIZED
Bus request level: BR 0
Release type: RWD
Requester type: FAIR
Timeout: T 50 us
FIFO mode: Yes [✓ No]
Status: Success

Fig. 5.2: The CAEN VME Settings Menu

I/O Setting Menu – Pulser

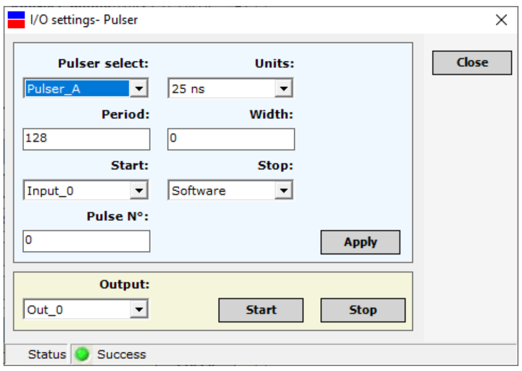
NOT SUPPORTED BY THE V3718/V4718 BRIDGE

The Pulser Setting Menu allows the performing of the settings of the V2718 and V1718 built-in pulsers described in detail in the Pulser sections of the Bridge User Manual**[RD4][RD5]**. The Bridge features two internal pulsers, called Pulser A and Pulser B

The output pulses are provided in the following way:

- Out_0 or Out_1 for Pulser A;
- Out_2 or Out_3 for Pulser B.

The programmable parameters are the step units (Units), the Period, the Width, and the number of produced pulses (Pulse N°). Start options are via software, via the SYSRES button (short pressure), or the Input_0/Input_1 signals. Stop options are either via software or via Input_0 (Pulser A) and Input_1 (Pulser B). The pulsers can be reset via the front panel SYSRES button **[RD4][RD5]**. Refer also to the “Input Multiplexer Set” register description in the Bridge User Manual.



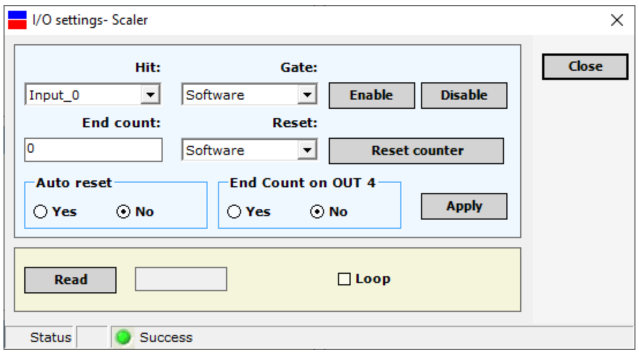
Pulser select: Pulser_A
Units: 25 ns
Period: 128
Width: 0
Start: Input_0
Stop: Software
Pulse N°: 0
Output: Out_0
Status: Success

Fig. 5.3: The I/O Setting Menu – Pulser

I/O Setting Menu – Scaler

NOT SUPPORTED BY THE V3718/V4718 BRIDGE

The Scaler Setting Menu allows the performing of the settings of the V2718 and V1718 built-in scaler described in detail in the relevant Bridge User Manual**[RD4][RD5]**. The Bridge features an internal scaler that counts hits arriving on the enabled front panel input (Hit = Input_0 or Input_1). Gate and Reset signals can be sent either on the unused input connector or software generated; an End Count pulse is eventually available on Out_4. Auto-reset and Loop options can be either enabled or disabled independently. It is possible to read the stored hits in the lower part of the Menu (Read). Refer also to the "Scaler 0" register description in the Bridge User Manual.


I/O settings- Scaler

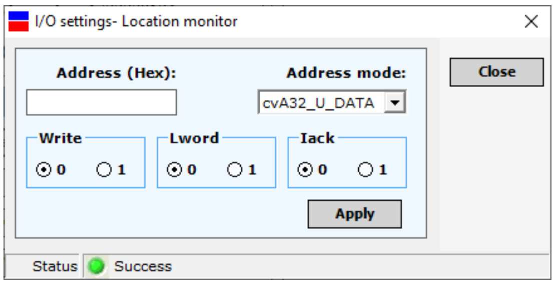
Hit: Input_0
Gate: Software
End count: 0
Reset: Software
Auto reset: Yes [✓ No]
End Count on OUT 4: Yes [✓ No]
Loop: ☐
Status: Success

Fig. 5.4: The I/O Setting Menu – Scaler

I/O Setting Menu – Location Monitor

NOT SUPPORTED BY THE V3718/V4718 BRIDGE

The Location Monitor Setting Menu allows producing an output signal when a particular VME cycle, at a particular base address, is detected. Refer also to the "Local Monitor" section in the Bridge User Manual**[RD4][RD5]**.


I/O settings- Location monitor

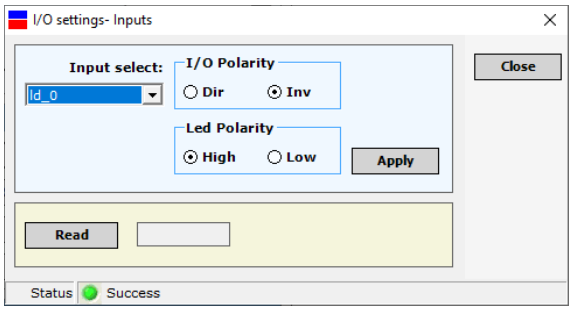
Address (Hex):
Address mode: cvA32_U_DATA
Write: [✓ 0] 1
Lword: [✓ 0] 1
Iack: [✓ 0] 1
Status: Success

Fig. 5.5: The I/O Setting Menu – Local Monitor

I/O Setting Menu – Inputs

NOT SUPPORTED BY THE V3718/V4718 BRIDGE

The Input Setting Menu allows setting the polarity of Input_0, Input_1, and of the relevant LEDs. Refer also to the "Input Multiplexer Set" and "LED Polarity Set" register descriptions in the Bridge User Manual**[RD4][RD5]**.


I/O settings- Inputs

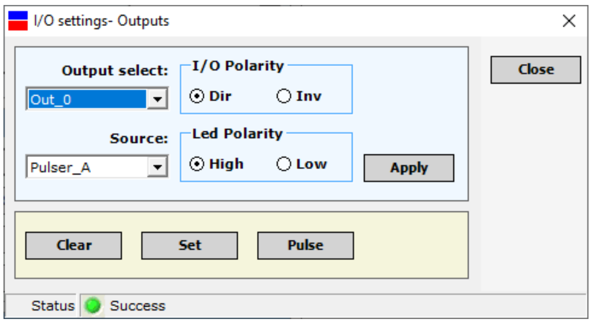
Input select: Id_0
I/O Polarity: Dir [✓ Inv]
Led Polarity: [✓ High] Low
Status: Success

Fig. 5.6: The I/O Setting Menu – Inputs

I/O Setting Menu – Outputs

NOT SUPPORTED BY THE V3718/V4718 BRIDGE

The Output Setting Menu allows setting the polarity of Output [0:4] and the relevant LEDs, as well as selecting the output source and producing an output pulse at will. Refer also to the "Output Multiplexer Set" register description in the Bridge User Manual**[RD4][RD5]**.



Output select: Out_0

Source: Pulser_A

I/O Polarity: [✓ Dir] Inv

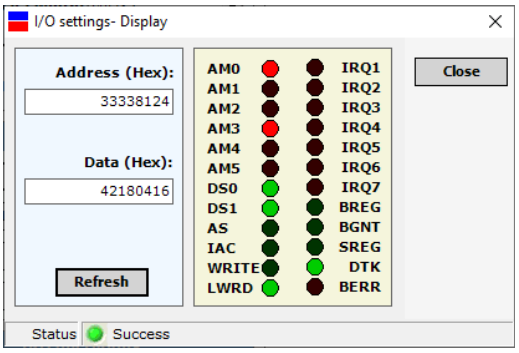
Led Polarity: [✓ High] Low

Status: Success

Fig. 5.7: The I/O Setting Menu – Outputs

I/O Setting Menu – Display

The Display Setting Menu allows monitoring the status of the Display corresponding to a serviced cycle. Refer also to the "Display Address Low" and "Display Control Right" register descriptions in the Bridge User Manual**[RD4][RD5]**.



Address (Hex): 33338124

Data (Hex): 42180416

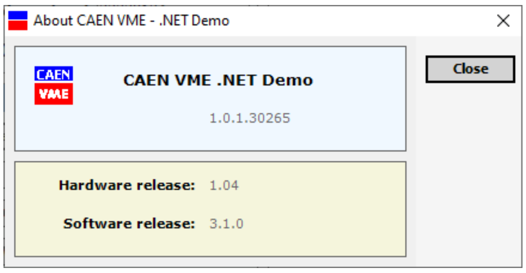
AM0	🔴	●	IRQ1
AM1	●	●	IRQ2
AM2	●	●	IRQ3
AM3	🔴	●	IRQ4
AM4	●	●	IRQ5
AM5	●	●	IRQ6
DS0	🟢	●	IRQ7
DS1	🟢	●	BREG
AS	●	●	BGNT
IAC	●	●	SREG
WRITE	●	🟢	DTK
LWRD	🟢	●	BERR

Status: Success

Fig. 5.8: The I/O Setting Menu – Display

I/O Setting Menu – About

The About Setting Menu informs on the revision number of the running software and hardware firmware.



CAEN VME .NET Demo

1.0.1.30265

Hardware release: 1.04

Software release: 3.1.0

Fig. 5.9: The I/O Setting Menu – About

6 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/

Scan me

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

UM7715 - CAENVMELib User & Reference Manual rev. 2 - September 1st, 2021
00101/18:VMELib.MUTX/01