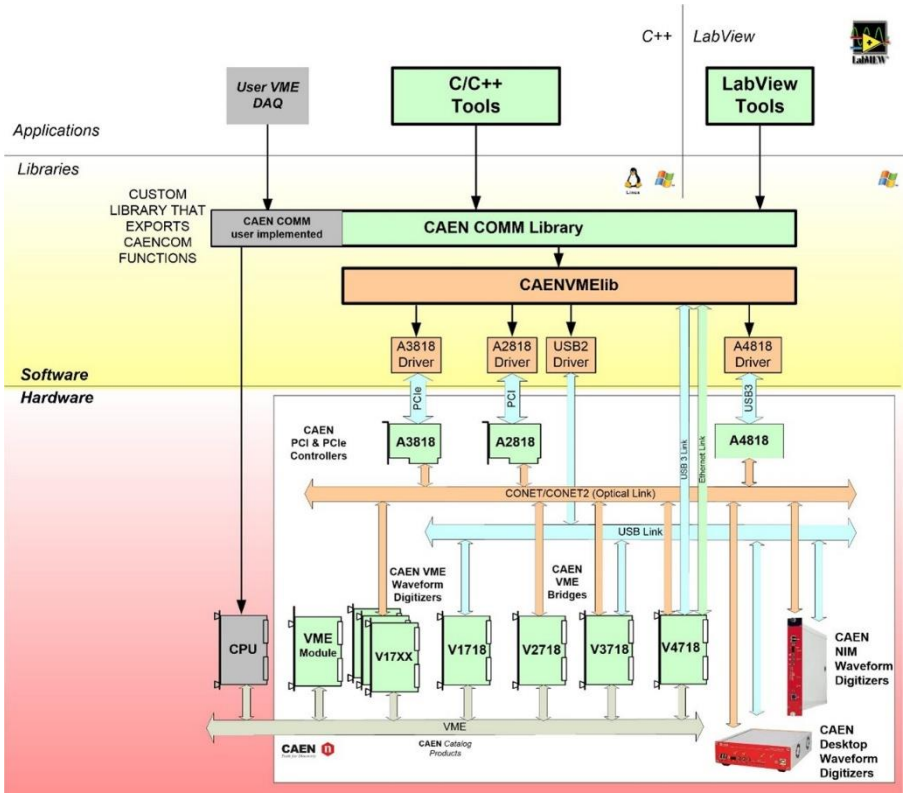
Rev. 6 - June 27th, 2024

CAENVMELib Library

Interface library for CAEN VME Bridges



CAEN Tools for Discovery

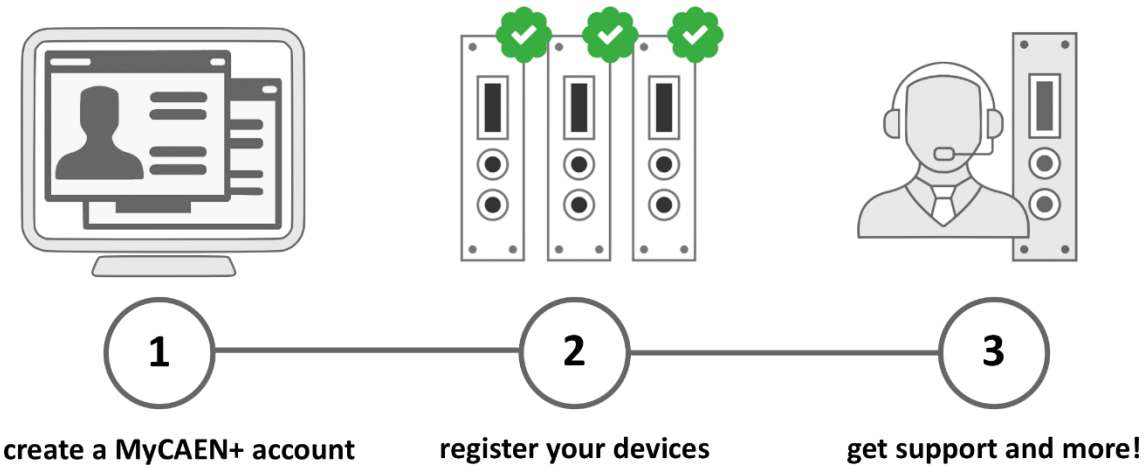
User Manual

UM7715

Register your device

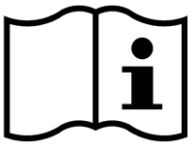
Register your device to your **MyCAEN+** account and get access to our customer services, such as notification for new firmware or software upgrade, tracking service procedures or open a ticket for assistance. **MyCAEN+** accounts have a dedicated support service for their registered products. A set of basic information can be shared with the operator, speeding up the troubleshooting process and improving the efficiency of the support interactions.

MyCAEN+ dashboard is designed to offer you a direct access to all our after sales services. Registration is totally free, to create an account go to https://www.caen.it/become-mycaenplus-user and fill the registration form with your data.



https://www.caen.it/become-mycaenplus-user/

Purpose of this User Manual



This User Manual contains the full description of the CAENVMELib library, including software installation instructions, a description of the C functions and provided demos, a brief representation of the relevant LabVIEW VIs.

Change Document Record

Date	Revision	Changes
September 18th, 2020	00	First release*
May 17th, 2021	01	Added A4818 support
September 1st, 2021	02	Added V4718 support and CAENVME_Init2, CAENVME_DecodeError, CAENVME_DriverRelease, CAENVME_FIFOMBLTReadCycle functions
January 25th, 2022	03	Added Chap. **6** in preplacement of the previous section on CAENVMELib Demos. Added notes in Sections **5.2.16**, **5.3.2**, **0**, **5.3.4**, **0**. Added new scaler sections from **5.5.3** to **5.5.25**. Updated each C function by adding the relevant LabVIEW VI description.
January 24 th, 2023	04	CAENVME_Init marked as deprecated and added note that warns users that arguments have been swapped since patch 3.4.0.
February 7th, 2024	05	Added connection parameter for A5818.
June 27th, 2024	06	Added CAENVME Python Binding chapter.
*Previous documentation on CAENVMELib used to be included in the V1718 and V2718 CAEN Bridge user manuals.		

Symbols, Abbreviated Terms, and Notations

DLL	Dynamic-Link Library
IRQ	Interrupt Request
PID	Product Identifier

Reference Documents

[RD1] AN2472 – CONET1 to CONET2 migration

[RD2] UM7685 – V3718 User Manual

[RD3] V1718 User Manual

[RD4] V2718 User Manual

[RD5] A2818 User Manual

[RD6] A3818 User Manual

[RD7] DS7799 – A4818 Data Sheet

[RD8] UM8305 – V4718 User Manual

https://www.caen.it/support-services/documentation-area/

Manufacturer Contacts

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

Index

Purpose of this User Manual 2
Change Document Record 2
Symbols, Abbreviated Terms, and Notations 2
Reference Documents 2
Manufacturer Contacts 3
Index 4
List of Figures 6
List of Tables 6
1 Introduction 7
2 System Requirements 8
2.1 Software 8
2.2 Hardware 8
3 CAENVMELib Installation 9
3.1 Windows OS 9
3.2 Linux OS 10
4 CAENVMELib Return Codes 11
5 CAENVMELib Functions 12
5.1 Generic Functions 12
5.1.1 CAENVME_Init (Deprecated) 12
5.1.2 CAENVME_Init2 13
5.1.3 CAENVME_End 14
5.1.4 CAENVME_DecodeError 14
5.1.5 CAENVME_BoardFWRelease 14
5.1.6 CAENVME_SWRelease 15
5.1.7 CAENVME_DriverRelease 15
5.1.8 CAENVME_DeviceReset 15
5.1.9 CAENVME_ReadRegister 16
5.1.10 CAENVME_WriteRegister 16
5.2 VME Communication Functions 17
5.2.1 CAENVME_ReadCycle 17
5.2.2 CAENVME_WriteCycle 17
5.2.3 CAENVME_MultiRead 18
5.2.4 CAENVME_MultiWrite 19
5.2.5 CAENVME_BLTReadCycle 20
5.2.6 CAENVME_RMWCycle 21
5.2.7 CAENVME_MBLTReadCycle 21
5.2.8 CAENVME_BLTWriteCycle 22
5.2.9 CAENVME_MBLTWriteCycle 23
5.2.10 CAENVME_FIFOBLTReadCycle 24
5.2.11 CAENVME_FIFOBLTWriteCycle 25
5.2.12 CAENVME_FIFOMBLTReadCycle 26
5.2.13 CAENVME_FIFOMBLTWriteCycle 27
5.2.14 CAENVME_ADOCycle 27
5.2.15 CAENVME_ADOHCycle 28
5.2.16 CAENVME_SetArbiterType 28
5.2.17 CAENVME_GetArbiterType 29
5.2.18 CAENVME_SetRequesterType 29
5.2.19 CAENVME_GetRequesterType 30
5.2.20 CAENVME_SetReleaseType 30
5.2.21 CAENVME_GetReleaseType 31
5.2.22 CAENVME_SetBusReqLevel 31
5.2.23 CAENVME_GetBusReqLevel 32
5.2.24 CAENVME_SetTimeout 32
5.2.25 CAENVME_GetTimeout 33
5.2.26 CAENVME_SetFIFOMode 33

5.2.27 CAENVME_GetFIFOMode 34
5.2.28 CAENVME_ReadDisplay 34
5.2.29 CAENVME_SetLocationMonitor 35
5.2.30 CAENVME_SystemReset 35
5.2.31 CAENVME_BLTReadAsync 36
5.2.32 CAENVME_BLTReadWait 36
5.3 Interrupt Control Functions 37
5.3.1 CAENVME_IACKCycle 37
5.3.2 CAENVME_IRQCheck 37
5.3.3 CAENVME_IRQEnable 38
5.3.4 CAENVME_IRQDisable 38
5.3.5 CAENVME_IRQWait 39
5.4 Pulser Control Functions 40
5.4.1 CAENVME_SetPulserConf 40
5.4.2 CAENVME_StartPulser 41
5.4.3 CAENVME_GetPulserConf 42
5.4.4 CAENVME_StopPulser 43
5.5 Scaler Control Functions 44
5.5.1 CAENVME_SetScalerConf 44
5.5.2 CAENVME_GetScalerConf 45
5.5.3 CAENVME_ResetScalerCount 45
5.5.4 CAENVME_EnableScalerGate 46
5.5.5 CAENVME_DisableScalerGate 46
5.5.6 CAENVME_SetScaler_Mode 46
5.5.7 CAENVME_GetScaler_Mode 47
5.5.8 CAENVME_SetScaler_InputSource 47
5.5.9 CAENVME_GetScaler_InputSource 48
5.5.10 CAENVME_SetScaler_GateSource 48
5.5.11 CAENVME_GetScaler_GateSource 49
5.5.12 CAENVME_SetScaler_ClearSource 49
5.5.13 CAENVME_SetScaler_StartSource 50
5.5.14 CAENVME_GetScaler_StartSource 50
5.5.15 CAENVME_SetScaler_ContinuousRun 51
5.5.16 CAENVME_GetScaler_ContinuousRun 51
5.5.17 CAENVME_SetScaler_MaxHits 52
5.5.18 CAENVME_GetScaler_MaxHits 52
5.5.19 CAENVME_SetScaler_DWellTime 53
5.5.20 CAENVME_GetScaler_DWellTime 53
5.5.21 CAENVME_SetScaler_SWStart 54
5.5.22 CAENVME_SetScaler_SWStop 54
5.5.23 CAENVME_SetScaler_SWReset 54
5.5.24 CAENVME_SetScaler_SWOpenGate 55
5.5.25 CAENVME_SetScaler_SWCloseGate 55
5.6 I/O Control Functions 56
5.6.1 CAENVME_SetOutputConf 56
5.6.2 CAENVME_GetOutputConf 57
5.6.3 CAENVME_SetOutputRegister 57
5.6.4 CAENVME_ClearOutputRegister 58
5.6.5 CAENVME_PulseOutputRegister 58
5.6.6 CAENVME_SetInputConf 59
5.6.7 CAENVME_GetInputConf 59
6 CAENVME Python Binding 60
7 CAENVME Demo 61
7.1 Run the CAENVME Demo 61
7.1.1 WINDOWS 61
7.1.2 LINUX 61
7.1.3 CONNECTION PARAMETERS 61
7.1.4 DEMO CONSOLE DESCRIPTION 62
7.2 Examples 63
7.2.1 Single Read Example 63
7.2.2 Single Write Example 63
7.2.3 Block Transfer Read Example 65
7.2.4 VSL Script Function 67

8 Technical Support 70

List of Figures

Fig. 1.1: Hardware and Software layers 7
Fig. 3.1: License Agreement step 9
Fig. 3.2: Select Destination Location step 9
Fig. 3.3: Select Components step 9
Fig. 3.4: Select Start Menu Folder step 9
Fig. 3.5: Start Installation step 10
Fig. 3.6: Completing Installation step 10
Fig. 6.1: CAENVME demo console usage message 61
Fig. 6.2: CAENVME demo console menu 62
Fig. 6.3: Read of a single register using the CAENVME demo console 63
Fig. 6.4: Single write, step 1 64
Fig. 6.5: Single write, step 2 64
Fig. 6.6: Single write, step 3 64
Fig. 6.7: Block transfer read, step 1 65
Fig. 6.8: Block transfer read, step 2 65
Fig. 6.9: Block transfer read, step 3 66
Fig. 6.10: Block transfer read, step 4 66
Fig. 6.11: Example of vsl script 69

List of Tables

Tab. 2.1: Software requirements 8
Tab. 2.2: Hardware requirements 8
Tab. 4.1: Return codes table 11
Tab. 5.1: Source selection table for the output lines for V1718/VX1718 and V2718/VX2718 56
Tab. 5.2: Source selection table for the output lines for V3718/VX3718 56
Tab. 6.1: Predefined variables 67
Tab. 6.2: Operators tab 67
Tab. 6.3: Conditions tab 68
Tab. 6.4: Commands tab 69

1 Introduction

The 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 as well as the A4818 adapter.

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

THE CAENVME REV 4.0.2 OR HIGHER IS REQUIRED TO OPERATE WITH THE A5818 CONTROLLER

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

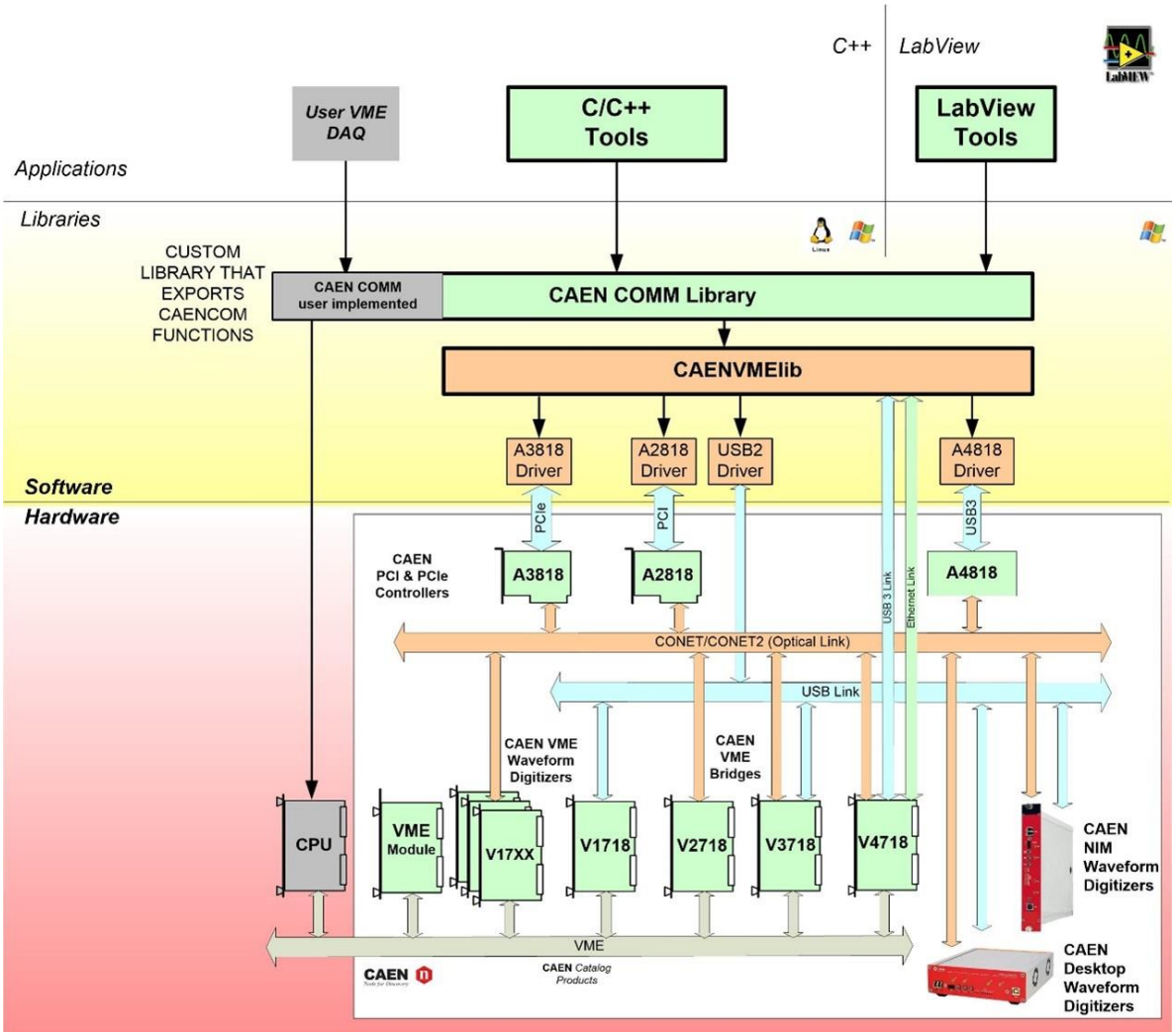


Fig. 1.1: Hardware and Software layers

Please, refer to CAEN Application Note **[RD1]** for details on CONET (CAEN Optical Link proprietary protocol) and migration from a CONET1 to a CONET2 system.

2 System Requirements

2.1 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™ rev. 8.2 or higher (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.

2.2 Hardware

Communication Mode	CAEN Hardware	CAEN Driver (Windows/Linux)
USB2	V1718	V1718 USB driver
	V3718	V3718 USB driver
CONET -> VME	A2818, A3818, or A5818 and V2718, V3718 or V4718	A2818, A3818, or A5818 CONET driver
USB3 -> CONET -> VME	A4818 and V2718, V3718 or V4718	A4818 USB driver (Windows only)
USB3	V4718	V4718 USB driver (Linux only)
ETHERNET	V4718	No drivers required

Tab. 2.2: Hardware requirements

For the complete hardware description of CAEN Bridges, Controllers and Adapter, please refer to the relevant user documentation **[RD2][RD3][RD4][RD5][RD6][RD7][RD8]**.

3 CAENVMELib Installation

To install the CAENVMELib library, follow the steps below:

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

3.1 Windows OS

The procedure is based on a 64-bit Windows 10 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**).

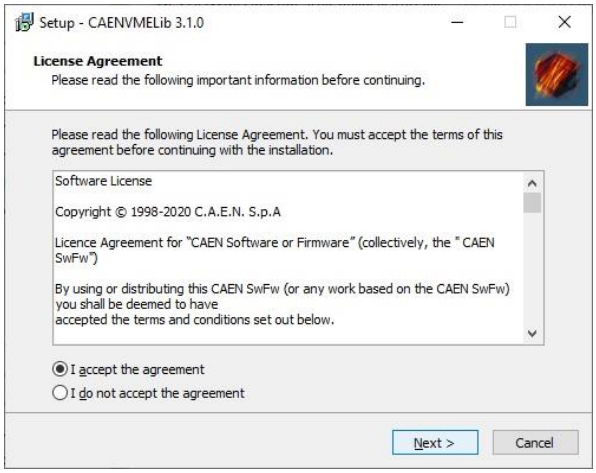


Fig. 3.1: License Agreement step

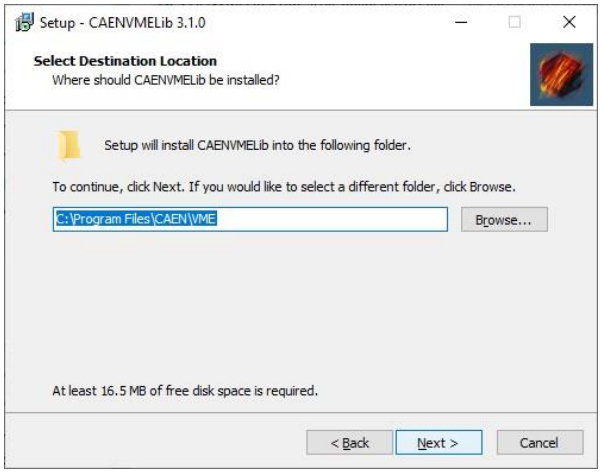


Fig. 3.2: Select Destination Location step

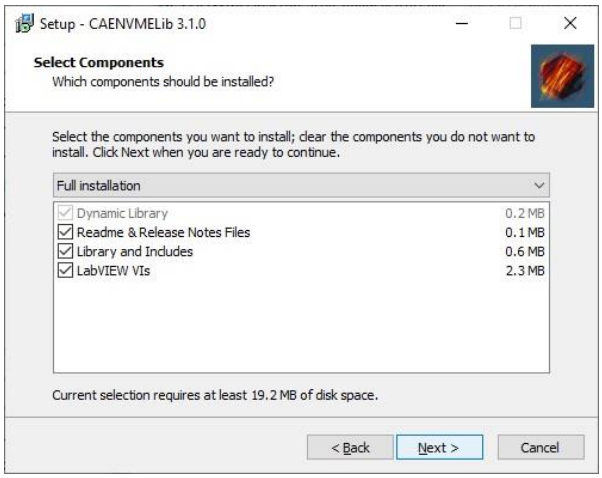


Fig. 3.3: Select Components step

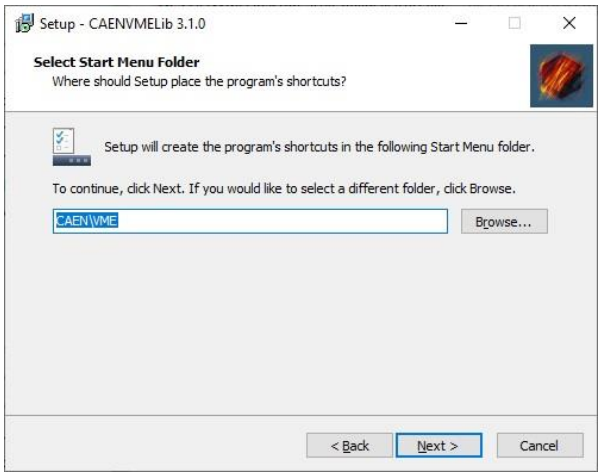


Fig. 3.4: Select Start Menu Folder step

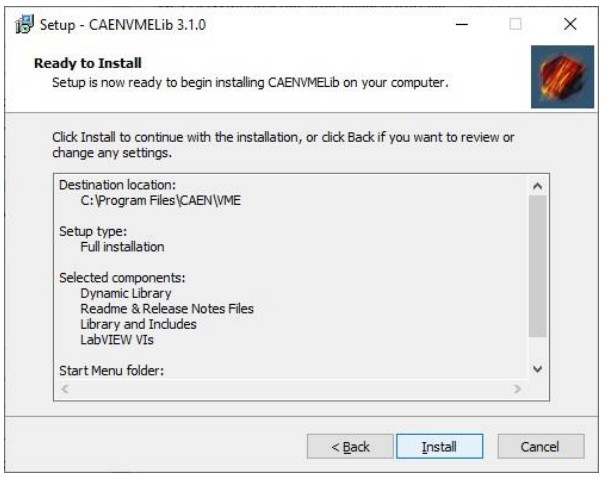


Fig. 3.5: Start Installation step

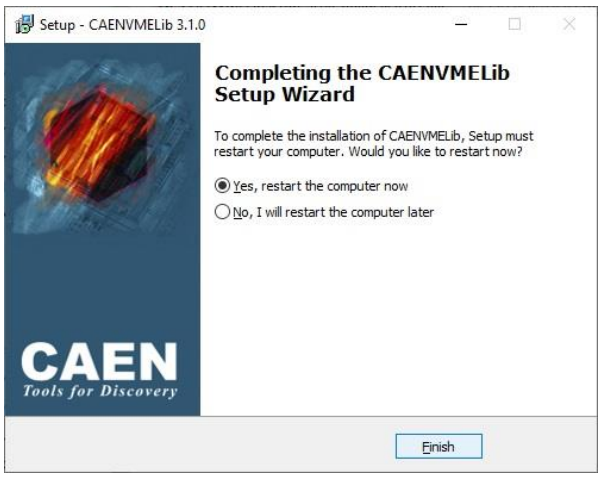


Fig. 3.6: Completing Installation step

3.2 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 Return Codes

Every function of the CAENVMELib returns a CVErrorCodes type (see definition in CAENVMEtypes.h). **Tab. 4.1** shows the meaning of the codes returned by those functions.

Code	Value	Description
cvSuccess	0	Operation 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

5 CAENVMELib Functions

5.1 Generic Functions

5.1.1 CAENVME_Init (Deprecated)

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.

Note: Version 3.4.0 fix bug added in 3.2.0, which swapped the second and third argument of CAENVME_Init for all board types except for cvV1718, cvUSB_V3718 and cvUSB_V3718_LOCAL. All users that are upgrading from 3.2.0 or newer version, and that are not using one of these board types, must swap the second and third arguments of CAENVME_Init or, preferably, use CAENVME_Init2. Users of CAENComm library that also upgraded that library to CAENComm 1.4.1 did not notice any difference because internally the CAENComm started using the swapped arguments. This this change, the CAENVME Library is backward compatible with versions before 3.2.0. We strongly suggest users of CAENComm to upgrate to the latest version, 1.6.0, released contextually with this release. Also, all new users should use CAENVME_Init2 instead of CAENVME_Init, that is kept only for legacy support. Note also that the interface of CAENVME_Init2, that has been introduced on version 3.3.0, has not been modified.

Synopsis

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

Arguments

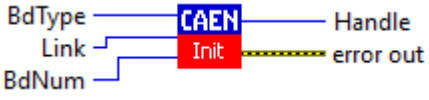
Name	Dir.	Description
BdType	in	Indicates the model of the bridge. Values can be: *cvV1718* (for the USB link with V1718 CAEN Bridge) *cvV2718* (for the CONET link with V2718 CAEN Bridge) *cvA2818* (for the CONET link with A2818 CAEN PCI Optical Controller) *cvA2719* (for the CONET link to A2719 mezzanine 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) 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)
***Handle**	out	Pointer to the handle that identifies the device.

Return Values

0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

CAENVME_Init.vi



5.1.2 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 types 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,
                        const void* arg,
                        short ConetNode,
                        int32_t *Handle
                        );
```

Arguments

Name	Dir.	Description
BdType	in	Indicates the model of the bridge. Values can be: *cvV1718* (USB to V1718) *cvV2718* (A2818/A3818/A5818 to V2718. Valid also for other CAEN bridges and direct link to digitizers) *cvA2818* (A2818 internal registers. Experts only) *cvA2719* (A2818/A3818/A5818 to A2719 internal registers. Experts only) *cvA3818* (A3818 internal registers. Experts only) *cvA5818* (A5818 internal registers. Experts only) *cvUSB_A4818_V2718_LOCAL* (A4818 to V2718 internal registers) *cvUSB_A4818_V2718* (A4818 to V2718) *cvUSB_A4818_LOCAL* (A4818 internal registers. Experts only) *cvUSB_A4818_V3718_LOCAL* (A4818 to V3718 internal registers) *cvUSB_A4818_V3718* (A4818 to V3718) *cvUSB_A4818* ((A4818 to CAEN digitizer) *cvUSB_A4818_A2719_LOCAL* (A4818 to A2719. Experts only) *cvUSB_V3718_LOCAL* (USB to V3718 internal registers) *cvPCI_A2818_V3718_LOCAL* (A2818 to V3718 internal registers) *cvPCIE_A3818_V3718_LOCAL* (A3818 to V3718 internal registers) *cvPCIE_A5818_V3718_LOCAL* (A5818 to V3718 internal registers) *cvUSB_V3718* (USB to V3718) *cvPCI_A2818_V3718* (A2818 to V3718) *cvPCIE_A3818_V3718* (A3818 to V3718) *cvPCIE_A5818_V3718* (A5818 to V3718) *cvUSB_V4718_LOCAL* (USB to V4718 internal registers. Experts only) *cvPCI_A2818_V4718_LOCAL* (A2818 to V4718 internal registers. Experts only) *cvPCIE_A3818_V4718_LOCAL* (A3818 to V4718 internal registers. Experts only) *cvPCIE_A5818_V4718_LOCAL* (A5818 to V4718 internal registers. Experts only) *cvETH_V4718_LOCAL* (Ethernet to V4718 internal registers. Experts only) *cvUSB_V4718* (USB to V4718) *cvPCI_A2818_V4718* (A2818 to V4718) *cvPCIE_A3818_V4718* (A3818 to V4718) *cvPCIE_A5818_V4718* (A5818 to V4718) *cvETH_V4718* (Ethernet to 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 a USB connection via V1718 or V3718 (uint32_t * type).* *Pointer to the optical link number, in case of an optical link connection via V2718, V3718, or V4718 (uint32_t * type).* *Pointer to the PID, in case of a USB connection to the A4818 or the V4718 (uint32_t * type).* *Null-terminated string to the IP address, in case of an Ethernet connection to the V4718 (char * type).*
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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_Init2.vi

5.1.3 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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_End.vi

5.1.4 CAENVME_DecodeError

Description
This function allows decoding an error code (see chapter **4**).

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.

LabVIEW Representation

CAENVME_DecodeError.vi

5.1.5 CAENVME_BoardFWRelease

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

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

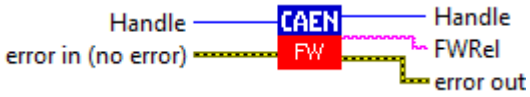
Arguments

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_BoardFWRel.vi



5.1.6 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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_SWRel.vi

5.1.7 CAENVME_DriverRelease

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

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_DriverRelease.vi

5.1.8 CAENVME_DeviceReset

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

Description
This function permits 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 **Tab. 4.1**).

LabVIEW Representation

n.a.

5.1.9 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
                                );
```

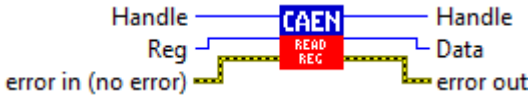
Arguments

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_ReadRegister.vi



5.1.10 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
                                 );
```

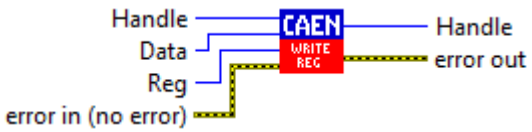
Arguments

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_WriteRegister.vi



5.2 VME Communication Functions

5.2.1 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
                                );
```

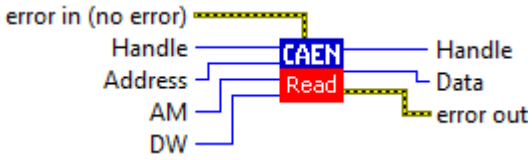
Arguments

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_ReadCycle.vi



5.2.2 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
                                );
```

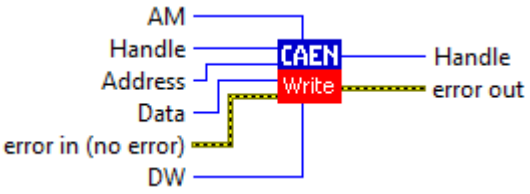
Arguments

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_WriteCycle.vi



5.2.3 CAENVME_MultiRead

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

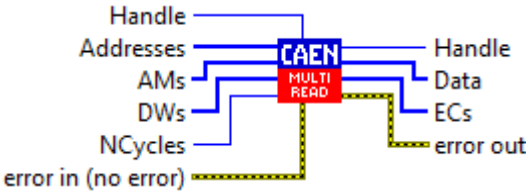
Name	Dir.	Description
handle	in	The handle that identifies the device.
Addrs	in	An array of VME bus addresses.
Buffer	out	An array of data which are read from the VME bus.
NCycles	in	The number of read cycles to perform.
AMs	in	An array of address modifiers (see CVAddressModifier enum in *CAENVMEtypes.h*).
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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_MultiRead.vi



5.2.4 CAENVME_MultiWrite

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
                                );
```

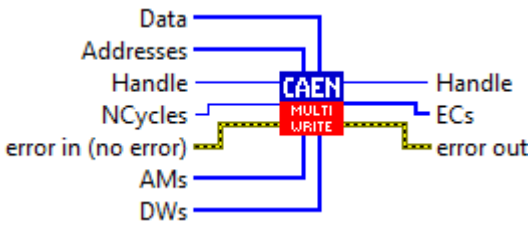
Arguments

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_MultiRead.vi



5.2.5 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
                                 );
```

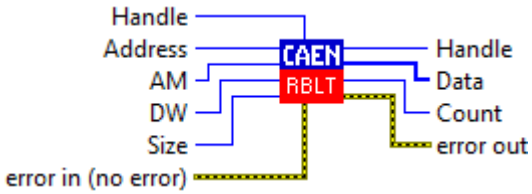
Arguments

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_BLTReadCycle.vi



5.2.6 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

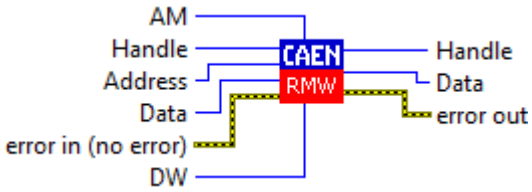
Name	Dir.	Description
Handle	in	The handle that identifies the device.
Address	in	The VME bus address.
Data	in/out	The data read and then written to the VME bus.
AM	in	The address modifier (see CVAddressModifier enum in *CAENVMEtypes.h*).
DW	in	The data width (see CVDataWidth enum in *CAENVMEtypes.h*).

Return Values

0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

CAENVME_RMWCycle.vi



5.2.7 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

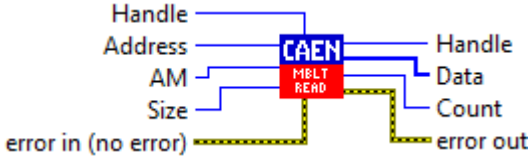
Name	Dir.	Description
Handle	in	The handle that identifies the device.
Address	in	The VME bus address.
Buffer	out	The data read from the VME bus.
Size	in	The size of the transfer in bytes.
AM	in	The address modifier (see CVAddressModifier enum in *CAENVMEtypes.h*).
count	out	The number of bytes transferred.

Return Values

0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

CAENVME_MBLTReadCycle.vi



5.2.8 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

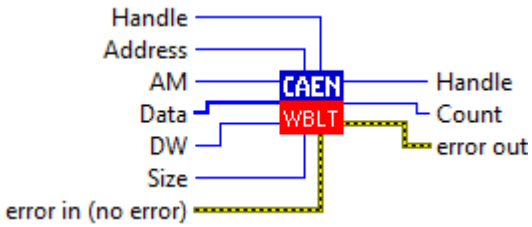
Name	Dir.	Description
Handle	in	The handle that identifies the device.
Address	in	The VME bus address.
Buffer	in	The data to be written to the VME bus.
Size	in	The size of the transfer in bytes.
AM	in	The address modifier (see CVAddressModifier enum in *CAENVMEtypes.h*).
DW	in	The data width (see CVDataWidth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

Return Values

0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

CAENVME_BLTWriteCycle.vi



5.2.9 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

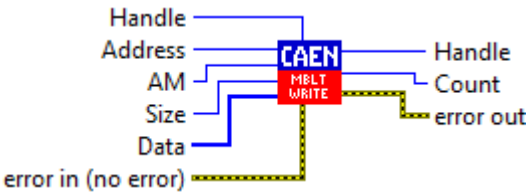
Name	Dir.	Description
Handle	in	The handle that identifies the device.
Address	in	The VME bus address.
Buffer	in	The data to be written to the VME bus.
Size	in	The size of the transfer in bytes.
AM	in	The address modifier (see CVAddressModifier enum in *CAENVMEtypes.h*).
count	out	The number of bytes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

Return Values

0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

CAENVME_MBLTReadCycle.vi



5.2.10CAENVME_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

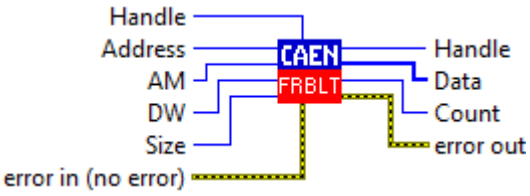
Name	Dir.	Description
Handle	in	The handle that identifies the device.
Address	in	The VME bus address.
Buffer	out	The data read from the VME bus.
Size	in	The size of the transfer in bytes.
AM	in	The address modifier (see CVAddressModifier enum in *CAENVMEtypes.h*).
DW	in	The data width (see CVDataWidth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

Return Values

0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

CAENVME_FIFOBLTReadCycle.vi



5.2.11 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

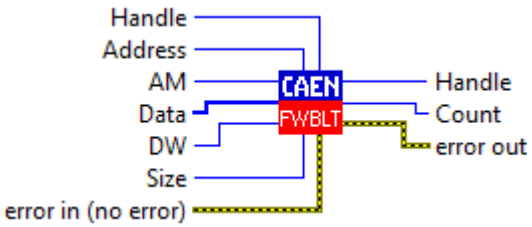
Name	Dir.	Description
Handle	in	The handle that identifies the device.
Address	in	The VME bus address.
Buffer	in	The data to be written to the VME bus.
Size	in	The size of the transfer in bytes.
AM	in	The address modifier (see CVAddressModifier enum in *CAENVMEtypes.h*).
DW	in	The data width (see CVDataWidth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

Return Values

0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

CAENVME_FIFOBLTWriteCycle.vi



5.2.12CAENVME_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

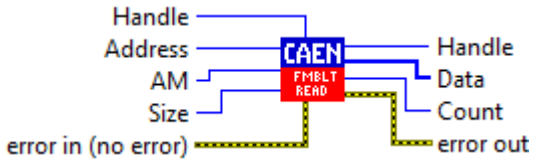
Name	Dir.	Description
Handle	in	The handle that identifies the device.
Address	in	The VME bus address.
Buffer	out	The data read from the VME bus.
Size	in	The size of the transfer in bytes.
AM	in	The address modifier (see CVAddressModifier enum in *CAENVMEtypes.h*).
count	in	The number of bytes transferred.

Return Values

0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

CAENVME_FIFOMBLTReadCycle.vi



5.2.13CAENVME_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

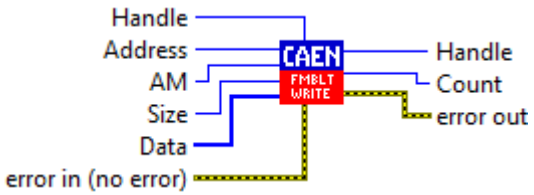
Name	Dir.	Description
Handle	in	The handle that identifies the device.
Address	in	The VME bus address.
Buffer	in	The data to be written to the VME bus.
Size	in	The size of the transfer in bytes.
AM	in	The address modifier (see CVAddressModifier enum in *CAENVMEtypes.h*).
count	out	The number of bytes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

Return Values

0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

CAENVME_FIFOMBLTWriteCycle.vi



5.2.14CAENVME_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

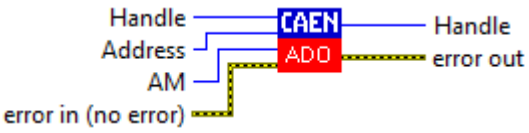
Name	Dir.	Description
Handle	in	The handle that identifies the device.
Address	in	The VME bus address.
AM	in	The address modifier (see CVAddressModifier enum in *CAENVMEtypes.h*).

Return Values

0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

CAENVME_ADOCycle.vi



5.2.15CAENVME_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
                                );
```

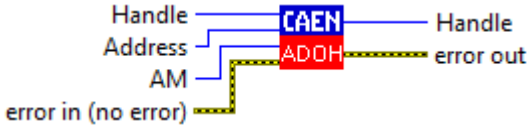
Arguments

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_ADOHCycle.vi



5.2.16CAENVME_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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_SetArbiterType.vi

5.2.17CAENVME_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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_GetArbiterType.vi

5.2.18CAENVME_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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_SetRequesterType.vi

5.2.19CAENVME_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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_GetRequesterType.vi

5.2.20CAENVME_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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_SetReleaseType.vi

5.2.21 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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_GetReleaseType.vi

5.2.22 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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_SetBusReqLevel.vi

5.2.23CAENVME_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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_GetBusReqLevel.vi

5.2.24CAENVME_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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_SetTimeout.vi

5.2.25CAENVME_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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_GetTimeout.vi

5.2.26CAENVME_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
                                );
```

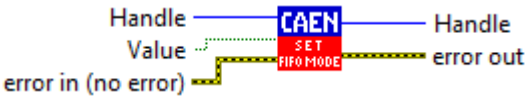
Arguments

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_SetFIFOMode.vi



5.2.27CAENVME_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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_GetFIFOMode.vi

5.2.28CAENVME_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
                                );
```

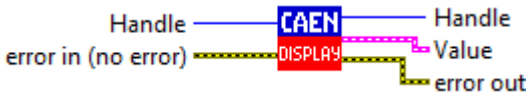
Arguments

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_ReadDisplay.vi



5.2.29CAENVME_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
                                       );
```

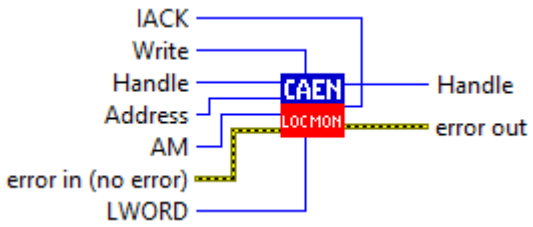
Arguments

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_SetLocationMonitor.vi



5.2.30CAENVME_SystemReset

Description
This function performs a system reset on the Bridge.

Synopsis

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

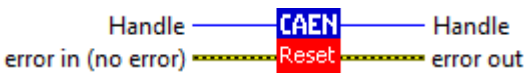
Arguments

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

Return Values
0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

CAENVME_SystemReset.vi



5.2.31CAENVME_BLTReadAsync

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

THIS FUNCTION IS NOT SUPPORTED BY THE A5818 PCIe CONET CONTROLLER

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 **Tab. 4.1**).

LabVIEW Representation

n.a.

5.2.32CAENVME_BLTReadWait

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

THIS FUNCTION IS NOT SUPPORTED BY THE A5818 PCIe CONET CONTROLLER

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 **Tab. 4.1**).

LabVIEW Representation

n.a.

5.3 Interrupt Control Functions

5.3.1 CAENVME_IACKCycle

Description
When using the CONET connection, this function performs a VME interrupt acknowledge cycle.

Note: This function cannot be used with USB link, including the A4818 Adapter.

Synopsis

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

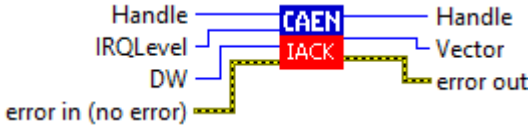
Arguments

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_IACKCycle.vi



5.3.2 CAENVME_IRQCheck

Description
When using the CONET connection, this function returns a bitmask indicating the active IRQ lines.

Note: This function cannot be used with USB link, including the A4818 Adapter.

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_IRQCheck.vi

5.3.3 CAENVME_IRQEnable

Description
When using the CONET connection, this function enables the IRQ lines specified by a mask.

Note: This function cannot be used with USB link, including the A4818 Adapter.

Synopsis

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

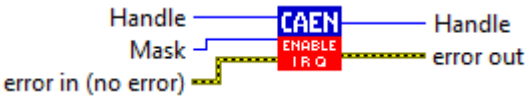
Arguments

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_IRQEnable.vi



5.3.4 CAENVME_IRQDisable

Description
When using the CONET connection, this function disables the IRQ lines specified by Mask.

Note: This function cannot be used with USB link, including the A4818 Adapter.

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_IRQDisable.vi

5.3.5 CAENVME_IRQWait

Description

When using the CONET connection, this function waits for the IRQ lines specified by the mask until one of them raises, or the timeout expires.

Note: This function cannot be used with USB link, including the A4818 Adapter.

Synopsis

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

Arguments

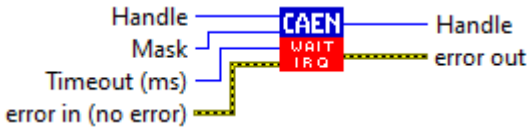
Name	Dir.	Description
Handle	in	The handle that identifies the device.
Mask	in	A bit-mask indicating the IRQ lines.
Timeout	in	Timeout in milliseconds.

Return Values

0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

CAENVME_IRQWait.vi



5.4 Pulser Control Functions

5.4.1 CAENVME_SetPulserConf

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.

Note: This function works only for V1718/VX1718, V2718/VX2718, and V3718/VX3718.

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
                                  );
```

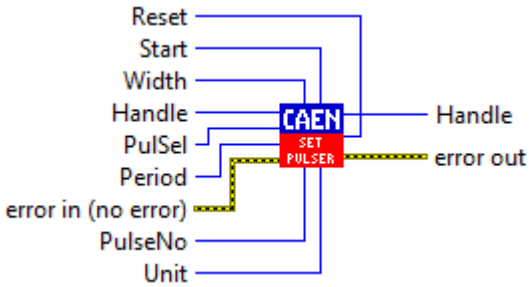
Arguments

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_SetPulserConf.vi



5.4.2 CAENVME_StartPulser

Description

This function starts the generation of the pulse burst if the specified pulser is configured for manual/software operation (see Sec. **5.4.1**).

Note: This function works only for V1718/VX1718, V2718/VX2718 and V3718/VX3718.

Synopsis

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

Arguments

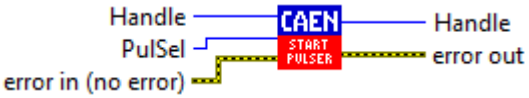
Name	Dir.	Description
Handle	in	The handle that identifies the device.
PulSel	in	The pulser to configure (see CVPulserSelect enum in *CAENVMEtypes.h*).

Return Values

0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

CAENVME_StartPulser.vi



5.4.3 CAENVME_GetPulserConf

Description
This function permits the reading of the pulsers configuration.

Note: This function works only for V1718/VX1718, V2718/VX2718, and V3718/VX3718.

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
                                  );
```

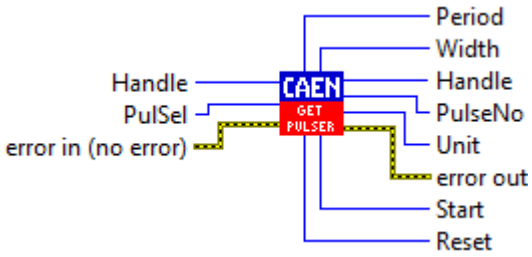
Arguments

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_GetPulserConf.vi



5.4.4 CAENVME_StopPulser

Description

This function stops the generation of the pulse burst if the specified pulser is configured for manual/software operation (see Sec. **5.4.1**).

Note: This function works only for V1718/VX1718, V2718/VX2718, and V3718/VX3718.

Synopsis

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

Arguments

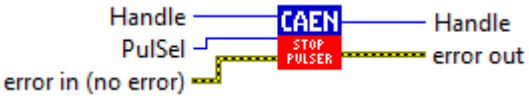
Name	Dir.	Description
Handle	in	The handle that identifies the device.
PulSel	in	The pulser to configure (see CVPulserSelect enum in *CAENVMEtypes.h*).

Return Values

0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

CAENVME_StopPulser.vi



5.5 Scaler Control Functions

5.5.1 CAENVME_SetScalerConf

Description
This function permits configuring the scaler embedded on the Bridge.

Note: This function works only for V1718/VX1718 and V2718/VX2718.

Synopsis

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

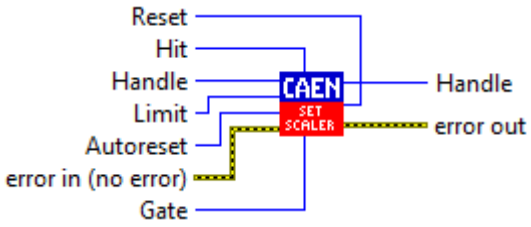
Arguments

Name	Dir.	Description
Handle	in	The handle that identifies the device.
Limit	in	The counter limit for the scaler (0 - 1023 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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_SetScalerConf.vi



5.5.2 CAENVME_GetScalerConf

Description
This function permits the reading of the scaler configuration.

Note: This function works only for V1718/VX1718 and V2718/VX2718.

Synopsis

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

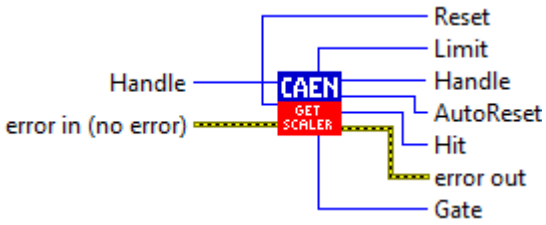
Arguments

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_GetScalerConf.vi



5.5.3 CAENVME_ResetScalerCount

Description
This function resets the counter of the scaler.

Note: This function works only for V1718/VX1718 and V2718/VX2718.

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_ResetScalerCount.vi

5.5.4 CAENVME_EnableScalerGate

Description
This function enables the gate of the scaler.

Note: This function works only for V1718/VX1718 and V2718/VX2718.

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_EnableScaelrGate.vi

5.5.5 CAENVME_DisableScalerGate

Description
This function disables the gate of the scaler.

Note: This function works only for V1718/VX1718 and V2718/VX2718.

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_DisableScalerGate.vi

5.5.6 CAENVME_SetScaler_Mode

Description
This function allows setting the scaler mode option on the Bridge.

Note: This function works only for V3718/VX3718.

Synopsis
```
CAENVME_API
CAENVME_GetScaler_Mode(
                       int32_t dev
                       CVScalerMode Mode
                       );
```

Arguments

Name	Dir.	Description
dev	in	The handle that identifies the device.
Mode	in	The mode option to set (see CVScalerMode enum in *CAENVMEtypes.h*).

Return Values
0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

n.a.

5.5.7 CAENVME_GetScaler_Mode

Description
This function allows getting the scaler mode configuration.

Note: This function works only for V3718/VX3718.

Synopsis

```
CAENVME_API
CAENVME_GetScaler_Mode(
                        int32_t dev,
                        CVScalerMode* Mode
                        );
```

Arguments

Name	Dir.	Description
dev	in	The handle that identifies the device.
Mode	out	The pointer to the scaler mode configuration to get (see CVScalerMode enum in *CAENVMEtypes.h*).

Return Values
0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

n.a.

5.5.8 CAENVME_SetScaler_InputSource

Description
This function allows setting the scaler input source option for the Bridge.

Note: This function works only for V3718/VX3718.

Synopsis

```
CAENVME_API
CAENVME_SetScaler_InputSource(
                               int32_t dev,
                               CVScalerSource Source
                               );
```

Arguments

Name	Dir.	Description
dev	in	The handle that identifies the device.
Source	out	The input source option to set (see CVScalerSource enum in *CAENVMEtypes.h*).

Return Values
0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

n.a.

5.5.9 CAENVME_GetScaler_InputSource

Description
This function allows getting the scaler input source configuration from the Bridge.

Note: This function works only for V3718/VX3718.

Synopsis

```
CAENVME_API
CAENVME_GetScaler_InputSource(
                                int32_t dev,
                                CVScalerSource* Source
                                );
```

Arguments

Name	Dir.	Description
dev	in	The handle that identifies the device.
Source	in	The pointer to the input source configuration to get (see CVScalerSource enum in *CAENVMEtypes.h*).

Return Values
0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

n.a.

5.5.10CAENVME_SetScaler_GateSource

Description
This function allows setting the scaler gate source on the Bridge.

Note: This function works only for V3718/VX3718.

Synopsis

```
CAENVME_API
CAENVME_SetScaler_GateSource(
                               int32_t dev,
                               CVScalerSource Source
                               );
```

Arguments

Name	Dir.	Description
dev	in	The handle that identifies the device.
Source	out	The input source option to set (see CVScalerSource enum in *CAENVMEtypes.h*).

Return Values
0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

n.a.

5.5.11 CAENVME_GetScaler_GateSource

Description
This function allows getting the scaler gate source configuration from the Bridge.

Note: This function works only for V3718/VX3718.

Synopsis

```
CAENVME_API
CAENVME_SetScaler_GateSource(
                              int32_t dev,
                              CVScalerSource* Source
                              );
```

Arguments

Name	Dir.	Description
dev	in	The handle that identifies the device.
Source	in	The pointer to the input source configuration to get (see CVScalerSource enum in *CAENVMEtypes.h*).

Return Values
0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

n.a.

5.5.12 CAENVME_SetScaler_ClearSource

Description
This function allows clearing the scaler source.

Note: This function works only for V3718/VX3718.

Synopsis

```
CAENVME_API
CAENVME_SetScaler_ClearSource(
                               int32_t dev,
                               CVScalerSource Source
                               );
```

Arguments

Name	Dir.	Description
dev	in	The handle that identifies the device.
Source	out	The pointer to the source to clear (see CVScalerSource enum in *CAENVMEtypes.h*).

Return Values
0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

n.a.

5.5.13CAENVME_SetScaler_StartSource

Description
This function allows setting the scaler start source.

Note: This function works only for V3718/VX3718.

Synopsis

```
CAENVME_API
CAENVME_SetScaler_StartSource(
                                int32_t dev,
                                CVScalerSource Source
                                );
```

Arguments

Name	Dir.	Description
dev	in	The handle that identifies the device.
Source	out	The start source to set (see CVScalerSource enum in *CAENVMEtypes.h*).

Return Values
0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

n.a.

5.5.14CAENVME_GetScaler_StartSource

Description
This function allows getting the scaler start source configuration.

Note: This function works only for V3718/VX3718.

Synopsis

```
CAENVME_API
CAENVME_GetScaler_StartSource(
                                int32_t dev,
                                CVScalerSource* Source
                                );
```

Arguments

Name	Dir.	Description
dev	in	The handle that identifies the device.
Source	in	The pointer to the start source configuration to get (see CVScalerSource enum in *CAENVMEtypes.h*).

Return Values
0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

n.a.

5.5.15CAENVME_SetScaler_ContinuousRun

Description
This function allows setting the scaler continuous run.

Note: This function works only for V3718/VX3718.

Synopsis

```
CAENVME_API
CAENVME_SetScaler_ContinuousRun(
                                int32_t dev,
                                CVContinuosRun OnOff
                                );
```

Arguments

Name	Dir.	Description
dev	in	The handle that identifies the device.
OnOff	out	The continuous run value to set (see CVContinuosRun enum in *CAENVMEtypes.h*).

Return Values
0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

n.a.

5.5.16CAENVME_GetScaler_ContinuousRun

Description
This function allows getting the scaler continuous run configuration.

Note: This function works only for V3718/VX3718.

Synopsis

```
CAENVME_API
CAENVME_GetScaler_ContinuousRun(
                                int32_t dev,
                                CVContinuosRun* OnOff
                                );
```

Arguments

Name	Dir.	Description
dev	in	The handle that identifies the device.
OnOff	in	The pointer to the continuous run configuration (see CVContinuosRun enum in *CAENVMEtypes.h*).

Return Values
0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

n.a.

5.5.17CAENVME_SetScaler_MaxHits

Description
This function allows setting the scaler MaxHits mode.

Note: This function works only for V3718/VX3718.

Synopsis

```
CAENVME_API
CAENVME_SetScaler_MaxHits(
                            int32_t dev,
                            uint16_t value
                            );
```

Arguments

Name	Dir.	Description
dev	in	The handle that identifies the device.
value	out	The MaxHits value (see CVContinuosRun enum in *CAENVMEtypes.h*).

Return Values
0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

n.a.

5.5.18CAENVME_GetScaler_MaxHits

Description
This function allows getting the scaler MaxHits value.

Note: This function works only for V3718/VX3718.

Synopsis

```
CAENVME_API
CAENVME_GetScaler_MaxHits(
                            int32_t dev,
                            uint16_t* value
                            );
```

Arguments

Name	Dir.	Description
dev	in	The handle that identifies the device.
value	in	The pointer to the MaxHits value to get

Return Values
0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

n.a.

5.5.19CAENVME_SetScaler_DWellTime

Description
This function allows setting the scaler D-Well time value.

Note: This function works only for V3718/VX3718.

Synopsis

```
CAENVME_API
CAENVME_SetScaler_DWellTime(
                            int32_t dev,
                            uint16_t value
                            );
```

Arguments

Name	Dir.	Description
dev	in	The handle that identifies the device.
value	out	The D-Well time value to set

Return Values
0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

n.a.

5.5.20CAENVME_GetScaler_DWellTime

Description
This function allows getting the scaler D-Well time value.

Note: This function works only for V3718/VX3718.

Synopsis

```
CAENVME_API
CAENVME_SetScaler_DWellTime(
                            int32_t dev,
                            uint16_t* value
                            );
```

Arguments

Name	Dir.	Description
dev	in	The handle that identifies the device.
value	in	The pointer to the D-Well time value to get

Return Values
0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

n.a.

5.5.21CAENVME_SetScaler_SWStart

Description
This function allows setting the scaler software start.

Note: This function works only for V3718/VX3718.

Synopsis
```
CAENVME_API
CAENVME_SetScaler_SWStart(
                            int32_t dev
                            );
```

Arguments

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

Return Values
0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

n.a.

5.5.22CAENVME_SetScaler_SWStop

Description
This function allows setting the scaler software stop.

Note: This function works only for V3718/VX3718.

Synopsis
```
CAENVME_API
CAENVME_SetScaler_SWStop(
                            int32_t dev
                            );
```

Arguments

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

Return Values
0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

n.a.

5.5.23CAENVME_SetScaler_SWReset

Description
This function allows setting the scaler software reset.

Note: This function works only for V3718/VX3718.

Synopsis
```
CAENVME_API
CAENVME_SetScaler_SWReset(
                            int32_t dev
                            );
```

Arguments

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

Return Values
0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

n.a.

5.5.24CAENVME_SetScaler_SWOpenGate

Description
This function allows setting the scaler software open gate.

Note: This function works only for V3718/VX3718.

Synopsis

```
CAENVME_API
CAENVME_SetScaler_ SWOpenGate (
                            int32_t dev
                            );
```

Arguments

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

Return Values
0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

n.a.

5.5.25CAENVME_SetScaler_SWCloseGate

Description
This function allows setting the scaler software close gate.

Note: This function works only for V3718/VX3718.

Synopsis

```
CAENVME_API
CAENVME_SetScaler_ SWCloseGate (
                            int32_t dev
                            );
```

Arguments

Name	Dir.	Description
dev	In	The handle that identifies the device.

Return Values
0: Success; Negative numbers are error codes (see **Tab. 4.1**).

LabVIEW Representation

n.a.

5.6 I/O Control Functions

5.6.1 CAENVME_SetOutputConf

Description
This function permits configuring the output lines of the Bridge.

Note: Works differently for V1718/VX1718/V2718/VX2718 and V3718/VX3718.

- **V1718/VX1718 and V2718/VX2718:** 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. 5.1** 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. 5.1: Source selection table for the output lines for V1718/VX1718 and V2718/VX2718

- **V3718/VX3718:** It is possible to specify the polarity for the line. The output line source depends by the *Source* parameter selected. **Tab. 5.2** shows the output configuration for every option.

		SOURCE SELECTION							
		cvVMESignals	cvCoincidence	cvInput0	cvInput1	cvPulserA	cvPulserB	cvScalerEnd	cvManualSW
OUTPUT	0	DS	Input Coinc.	In 0	In 1	Pulser A	Pulser B	Scaler End	Manual/SW
	1	AS	Input Coinc.	In 0	In 1	Pulser A	Pulser B	Scaler End	Manual/SW
	2	DTACK	Input Coinc.	In 0	In 1	Pulser A	Pulser B	Scaler End	Manual/SW
	3	BERR	Input Coinc.	In 0	In 1	Pulser A	Pulser B	Scaler End	Manual/SW

Tab. 5.2: Source selection table for the output lines for V3718/VX3718

Synopsis

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

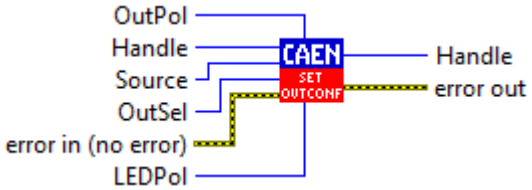
Arguments

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_SetOutputConf.vi



5.6.2 CAENVME_GetOutputConf

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

Note: Works only for V/VX1718, V/VX2718 and V3718/VX3718.

Synopsis

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

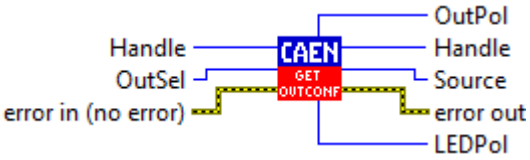
Arguments

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_GetOutputConf.vi



5.6.3 CAENVME_SetOutputRegister

Description
This function sets the specified lines.

Note: Works only for V/VX1718, V/VX2718 and V3718/VX3718.

Synopsis

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

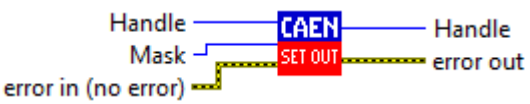
Arguments

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_SetOutputRegister.vi



5.6.4 CAENVME_ClearOutputRegister

Description
This function clears the specified lines.

Note: Works only for V/VX1718, V/VX2718, and V3718/VX3718.

Synopsis

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

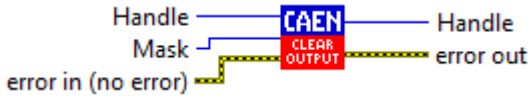
Arguments

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_ClearOutputRegister.vi



5.6.5 CAENVME_PulseOutputRegister

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

Note: Works only for V/VX1718, V/VX2718 and V3718/VX3718.

Synopsis

```
CAENVME API CAENVME PulseOutputRegister(
                                         int32_t Handle,
                                         unsigned short Mask
                                         );
```

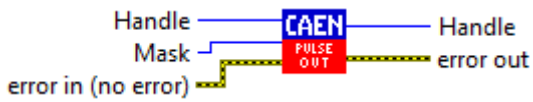
Arguments

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 (see **Tab. 4.1**).

LabVIEW Representation

CAENVME_PulseOutputRegister.vi



5.6.6 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.

Note: Works only for V/VX1718, V/VX2718 and V3718/VX3718.

Synopsis

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

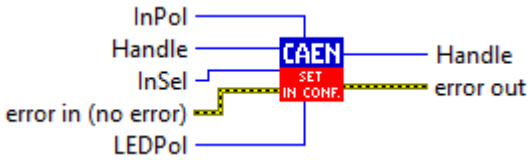
Arguments

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_SetInputConf.vi



5.6.7 CAENVME_GetInputConf

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

Note: Works only for V/VX1718, V/VX2718, and V3718/VX3718.

Synopsis

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

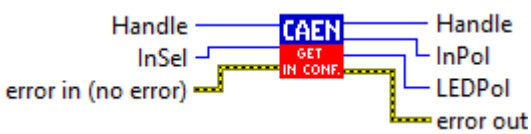
Arguments

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 **Tab. 4.1**).

LabVIEW Representation

CAENVME_GetInputConf.vi



6 CAENVME Python Binding

The CAENVME Python binding is distributed through the Pypi repository. It can be installed via the command:

pip install caen-libs

The package includes the CAENVME, the CAENComm, the CAEN PLU library and the CAEN HV Wrapper. It requires the C library installed. The CAENVME module can be import in the python script as follows:

```
from caen_libs import caenvme as vme
```

An example of the usage of the Python functions may be found on the CAEN GitHub repository (https://github.com/caenspa/py-caen-libs).

7 CAENVME Demo

In the package of the CAENVMELib, CAEN provides a simple C code, from now called CAENVME demo console, based on the functions of the library, to demonstrate how to control CAEN Bridges (V1718/VX1718, V2718/VX2718, V3718/VX3718, connected via A4818, USB, or CONET). For Users who want to develop their own software, the demo source files are included in the packet, providing a starting base for the customization.

The CAENVME demo console is described in the following section. The demo, as it is, manages only accesses to the VME Bus.

Before starting, make sure that your VME devices are properly installed.

7.1 Run the CAENVME Demo

7.1.1 WINDOWS

To run the CAENVME demo console on Windows:

- Open the Command Prompt
- Go into CAENVMELib installation folder (the default address is *C:\Program Files\CAEN\VME*).
- Enter in the "CAENVMEDemo" folder.
- Run the .exe file with the command *CAENVMEDemo.exe*.

7.1.2 LINUX

To run the CAENVME demo console on Linux:

- Open the terminal.
- Go into the CAENVMELib installation folder.
- Enter in the "sample" folder.
- Digit *make* to compile the executable.
- Run the executable file (*./CAENVMEDemo*).

7.1.3 CONNECTION PARAMETERS

Once the application is launched as explained before, a message indicates the correct syntax to run the CAENVME demo console (see **Fig. 7.1**).

```
C:\Program Files\CAEN\VME\CAENVMEDemo>CAENVMEDemo.exe
Usage: CAENVMEDemo V1718 <VMEDevice>
       CAENVMEDemo V2718 <VMEDevice> <VMELink>
       CAENVMEDemo USB_A4818 <PID>
```

Fig. 7.1: CAENVME demo console usage message

The defined commands to correctly launch the demo console are:

- *CAENVMEDemo V1718 <VMEDevice>* (to use in case of V1718/VX1718 or V3718/VX3718 connected via USB), where:

  <VMEDevice> is the board link number.

- *CAENVMEDemo V2718 <VMEDevice> <VMELink>* (to use in the case of V2718/VX2718 or V3718/VX3718 connected via CONET), where:

  <VMEDevice> is the board link number.

  <VMELink> is the CONET number in the daisy-chain loop.

- CAENVMEDemo USB_A4818 <PID> (to use in the case of connection via A4818), where:

  <PID> is the PID of the A4818.

As an example, to run the CAENVME demo console on Windows OS when connected through the USB link of the V3718 Bridge, the following command must be used:

C:\Program Files\CAEN\VME\CAENVMEDemo>CAENVMEDemo.exe V1718 0

7.1.4 DEMO CONSOLE DESCRIPTION

Launching the CAENVME demo console, the menu of the allowed key-commands appears (see **Fig. 7.2**).

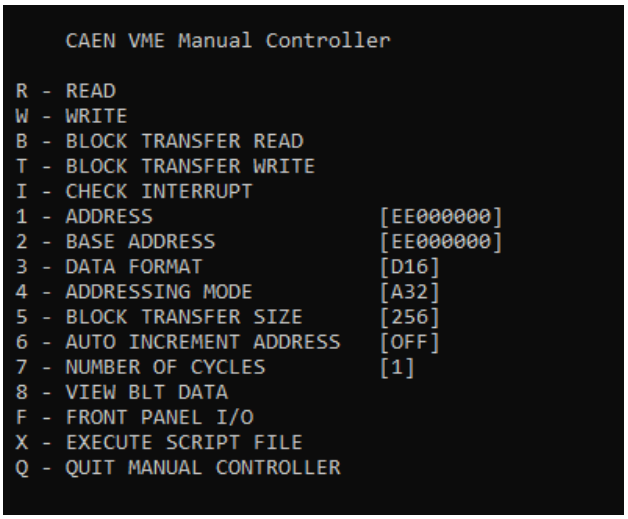


Fig. 7.2: CAENVME demo console menu

- R – READ: single read to register.
- W – WRITE: single write to register.
- B – BLOCK TRANSFER READ: reads in BLT mode.
- T – BLOCK TRANSFER WRITE: writes in BLT mode.sk
- I – CHECK INTERRUPT: checks the selected interrupt line on the VME bus.
- 1 – ADDRESS: register address of the VME Slave.
- 2 – BASE ADDRESS: VME base address of the Slave device.
- 3 – DATA FORMAT: sets the format of the data (both in reading and writing mode).
- 4 – ADDRESSING MODE: sets the address modifier.
- 5 – BLOCK TRANSFER SIZE: sets the dimension of the block transfer in bytes.
- 6 – AUTO INCREMENT ADDRESS: auto-increments the address on the VME bus.
- 7 – NUMBER OF CYCLES: number of cycles.
- 8 – VIEW BLT DATA: displays the transferred data.
- F – FRONT PANEL I/O: not implemented.
- X – EXECUTE SCRIPT FILE: executes a .vsl external script file (fully described in Sec. **7.2.4**); filename required to start.
- Q – QUIT MANUAL CONTROLLER: quits the demo console.

7.2 Examples

7.2.1 Single Read Example

This example shows how to make a read to get the ROC FPGA Firmware release of a CAEN VME digitizer (V/VX17xx) through a VME-to-CONET CAEN Bridge (V/VX2718 or V/VX3718).

- Run the demo console with “V2718 0 0” parameters.
- Type “2”, then enter the 8-digit hexadecimal VME Base address of the digitizer (in this example: 32100000).
- Type “1”, then enter the hexadecimal address of the ROC Firmware release register of the digitizer (i.e. 8124).
- Type “3” and set the digitizer data format (i.e. D32).
- Type “4” and set the addressing mode (digitizers accepts A24 and A32).
- Type “R” to read the register (the command is not key sensitive).
- The results appear as shown in **Fig. 7.3**. Refer to the documentation of the digitizer registers to decode the release number.

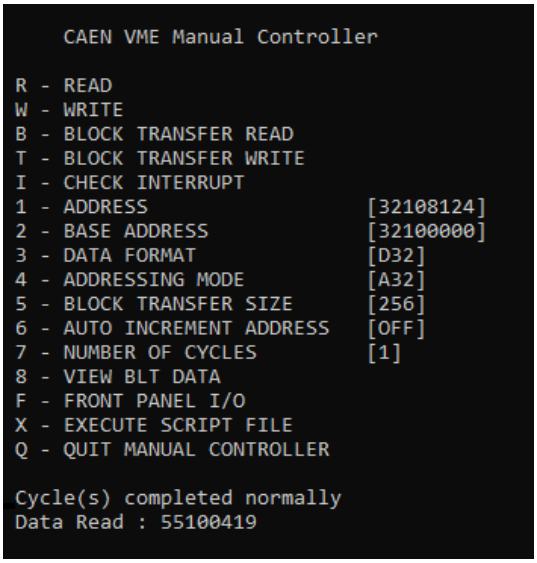


Fig. 7.3: Read of a single register using the CAENVME demo console

7.2.2 Single Write Example

This example shows how to make a write to disable the external trigger input of a CAEN VME MCA (V1782) through a VME-to-USB CAEN Bridge (V/VX1718 or V/VX3718).

- Run the demo console with “V1718 0” parameters.
- Type “2”, then enter the 8-digit hexadecimal VME Base Address of the digitizer (in this example: 32100000).
- Type “1”, then enter the hexadecimal address of the Disable External Trigger register of the MCA (i.e. 817C).
- Type “3” and set the digitizer data format (i.e. D32).
- Type “4” and set the correct addressing mode (the MCA accepts A24 and A32).
- Type “R” to read the register and check the status of the external trigger (**Fig. 7.4**); the default is enabled (0).

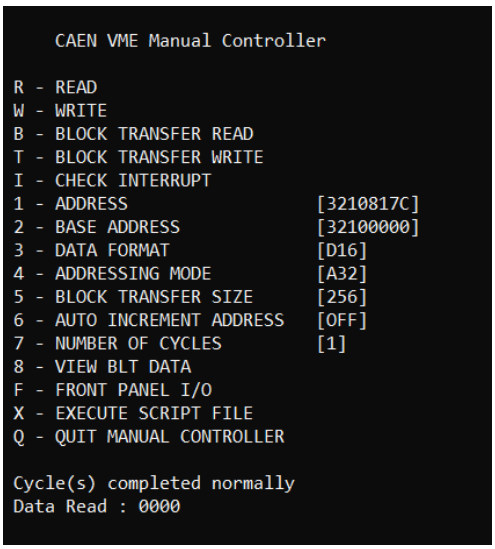


Fig. 7.4: Single write, step 1

- Type “W”, then digit 1 to disable the external trigger (**Fig. 7.5**).

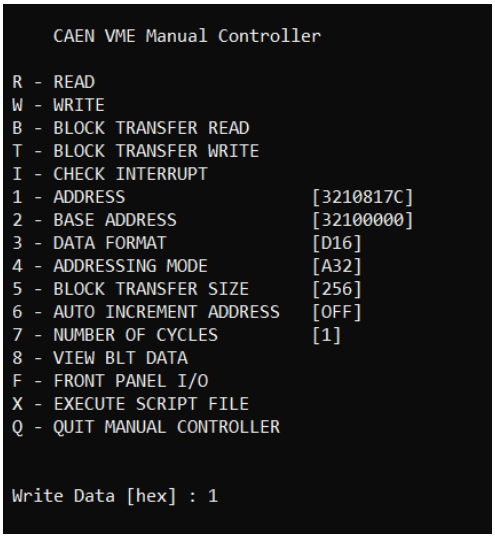


Fig. 7.5: Single write, step 2

- Type “R” again to check if the operation succeeded (**Fig. 7.6**).

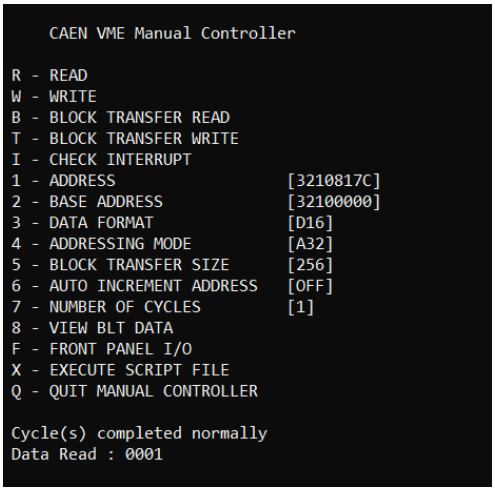


Fig. 7.6: Single write, step 3

7.2.3 Block Transfer Read Example

This example shows how to make a block transfer read to get the header of an event from the output buffer of a CAEN VME digitizer (V/VX17XX) through a VME-to-USB CAEN Bridge (V/VX1718 or V/VX3718).

To do that, some acquired data in the buffer are needed, so the digitizer must be put in run and then get triggers. For a complete understanding of the principles of operation, it is recommended to consult the digitizer and firmware registers documentation.

First step is to arm the digitizer acquisition logic:

- Run the demo console with “V1718 0” parameters.
- Type “2”, then enter the 8-digit hexadecimal VME Base Address of the digitizer (in this example: 32110000).
- Type “1”, then enter the hexadecimal address of the Acquisition Control register of the digitizer (i.e. 0x8100).
- Type “3” to set the data format (i.e. D32).
- Type “4” to set the addressing mode (i.e. A32).
- Type “W”, then digit 4 as the value to write (i.e. bit[2] = 1 arms the digitizer) as in **Fig. 7.7**. The RUN red Led on the digitizer front panel must light on.

```
    CAEN VME Manual Controller

R - READ
W - WRITE
B - BLOCK TRANSFER READ
T - BLOCK TRANSFER WRITE
I - CHECK INTERRUPT
1 - ADDRESS                 [32118100]
2 - BASE ADDRESS            [32110000]
3 - DATA FORMAT             [D32]
4 - ADDRESSING MODE         [A32]
5 - BLOCK TRANSFER SIZE     [256]
6 - AUTO INCREMENT ADDRESS  [OFF]
7 - NUMBER OF CYCLES        [1]
8 - VIEW BLT DATA
F - FRONT PANEL I/O
X - EXECUTE SCRIPT FILE
Q - QUIT MANUAL CONTROLLER

Cycle(s) completed normally
Write Data [hex] : 4
```

Fig. 7.7: Block transfer read, step 1

Next step is to send a software trigger:

- Type “1”, then enter the hexadecimal address of the Software Trigger register of the digitizer (i.e. 8108).
- Type “W”, then digit whichever hexadecimal value (e.g. “1”) to activate a single software trigger (**Fig. 7.8**). The TRG green Led on the digitizer front panel must flash. The event will then be stored in digitizer channel memory and available for the readout (the DRDY green Led must be on).

```
    CAEN VME Manual Controller

R - READ
W - WRITE
B - BLOCK TRANSFER READ
T - BLOCK TRANSFER WRITE
I - CHECK INTERRUPT
1 - ADDRESS                 [32118108]
2 - BASE ADDRESS            [32110000]
3 - DATA FORMAT             [D32]
4 - ADDRESSING MODE         [A32]
5 - BLOCK TRANSFER SIZE     [256]
6 - AUTO INCREMENT ADDRESS  [OFF]
7 - NUMBER OF CYCLES        [1]
8 - VIEW BLT DATA
F - FRONT PANEL I/O
X - EXECUTE SCRIPT FILE
Q - QUIT MANUAL CONTROLLER

Cycle(s) completed normally
Write Data [hex] : 1
```

Fig. 7.8: Block transfer read, step 2

Last step is to read out data from the digitizer. The output buffer of the board ranges between 0x0000 and 0x0FFC addresses. We suppose to read out just the header information of the event (refer to the digitizer User Manual), that is to say the fist four 32-bit words starting from the 0x0000 address

- Type “1”, then set the starting address of the digitizer event readout buffer (i.e. 0x0000).
- Type “5”, then set the block transfer size to the header dimension in the buffer in byte (i.e. 16).
- Type “6” to enable the auto-increment address (ON) and span the readout buffer space. The demo console settings are shown in **Fig. 7.9**.

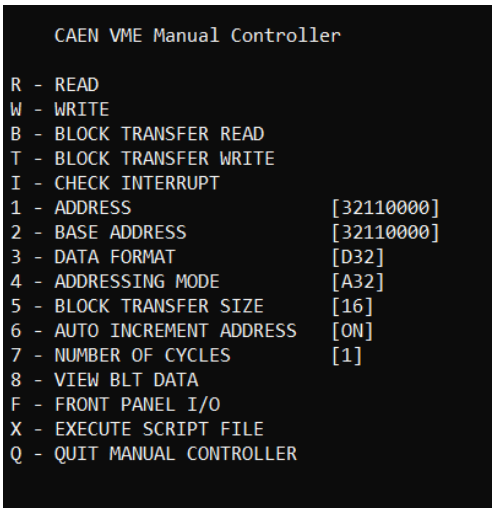


Fig. 7.9: Block transfer read, step 3

- Type “B” to perform the block transfer read.
- Type “8” to visualize the data read. The result is shown in **Fig. 7.10**.

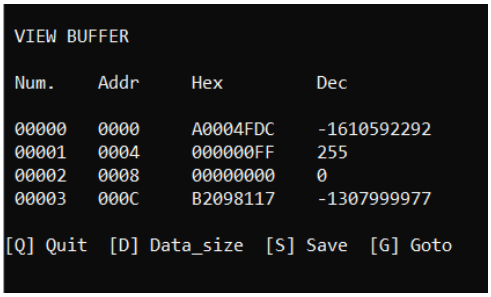


Fig. 7.10: Block transfer read, step 4

7.2.4 VSL Script Function

The CAENVME demo console allows the user to run a .vsl external script file. This must be written with a particular syntax based on the following allowed types:

- COMMENTS: all strings following "#" in line are comments.

- NUMBERS: numbers can be decimal or hexadecimal following the C notation.

  Example of usage:

  Var a *# defines "a" variable* (see the following bulleted point)

  a = 123 *# assigns a decimal value to "a"*

  a = 0x123 *# assigns a hexadecimal to "a"*

- VARIABLES: variables are defined as integers. The name of the variable is case-sensitive. There is a set of predefined variables described in **Tab. 7.1**.

Variable Name	Description
RData	The data read by the *READ* or *READREG* command (see the COMMANDS type)
BaseAddress	The Base Address for the VME access
Result	The result of the R/W access command (see the COMMANDS type): 0 = Success -1 = BusError -2 = Communication Error

Tab. 7.1: Predefined variables

- OPERATORS (variable assignment): operators and operands must be separated by spaces. The complete list is reported in **Tab. 7.2**.

Operator	Description	Example

=	Assigment	Var a # defines the "a" variable
		a = 0x0002 # assigns 2 to the variable
		a += 4 # adds 4 to the variable
		a <<= 2 # shifts the variable by 2 of bits to the left

+=	Addition	
-=	Subtraction	
*=	Multiplication	
/=	Division	
%=	Modulo	
&=	Bitwise AND	
|=	Bitwise OR	
^=	Bitwise Exclusive OR (XOR)	
<<=	Shift Left	
>>=	Shift Right	

Tab. 7.2: Operators tab

- CONDITIONS: (relational operators) must be included between round brackets "()"; operators and operands must be separated by spaces. The allowed operations are reported in **Tab. 7.3**.

Operator	Description	Example
==	Equal to	Var a # defines the "a" variable a &= 0x0001 # makes the bitwise AND between the variable and 0x0001 If (a != 0) Goto Label1 # jumps to Label1 if the variable is not equal to 0
<=	Less than or equal than	
>=	Grater than or equal than	
<	Less than	
>	Greater than	
!=	Not equal to	

Tab. 7.3: Conditions tab

- COMMANDS: commands are not case sensitive. The complete list of available commands is reported in .

Command	Description	Example
ALIAS	Text replacement	*Alias S1 S2* Replaces string S1 with S2
WRITE	VME write cycle. Requires to specify: - The Data Width (DW); admitted values are 1 = D8, 2 = D16, 4 = D32) - The Address Modifier (AM);the full list can be found in the library file *CAENVMETypes.h* - The Address Offset (AD); VME Address = Base Address + AD - The Data to write (DT) The result of the access is returned in the predefined variable; all parameters must be numbers or variable; expressions are not allowed	*Write DW AM AD DT*
READ	VME read cycle. Requires the same parameters as the WRITE. The Data read is assigned to the predefined variable *Rdata*. The result of the access is returned in the predefined variable *Result*; all parameters must be numbers or variables; expressions are not allowed	*Read DW AM AD DT*
WRITEREG	Write to Internal Register. Requires to specify: - The Register Address (AD) - The Data to write (DT) The result of the access is returned in the predefined variable *Result*; all parameters must be numbers or variables; expressions are not allowed	*WriteReg AD DT*
READREG	Read to internal register. Requires to specify: - The Register Address (AD) The Data read is assigned to the predefined variable Rdata. The result of the access is returned in the predefined variable *Result*; all parameters must be numbers or variables; expressions are not allowed	*ReadReg AD*
WAIT	Wait for a specific time T, expressed in milliseconds (ms)	*Wait T*
VAR	Variable definition. The variable (unsigned integer) is defined by the name V (V can be any string). Three predefined variablesa are available: - *VMEResult*: takes the result of the VME access - *BaseAddress*: base address for the VME access - *VMEData*: contains the data read on the bus with the Read operation	*Var V*
GOTO	Jump to a label. The labels are defined like in C (label:); labels can be any string.	*Goto Label*
RETURN	Return from jump. Returns to the instruction after the last "Goto"; nested "Goto-Return" commands are not allowed	*Return*
REPEAT – UNTIL	Conditional loop. Executes the commands between Repeat and Until until the condition is true; nested Repeat-Until commands are not allowed	*Repeat commands Until conditio]*

IF - ELSE	Conditional instructions. If the Condition is true, then executes Command1, else executes Command2; If and Else must be written in two consecutive lines	*If Condition Command1* *Else Command2*
PAUSE	Wait for a key. Stops the execution until a key is pressed	*Pause*
PRINT	Print of a message or a number to the Stdout Prints a string including one or two formatted values (C-like). The values can be numbers or variables. The strings are written to the Stdout and/or LogFile according to the relevant setting (see ECHO and LOG ON/OFF commands)	Print "String" value1 value2
ECHO	Echo to Stdout. Enables or disables the echo of the commands to the Stdout; default is OFF	*Echo ON* *Echo OFF*
LOG	Echo to LogFile. Enables or disables the echo of the commands to the Log File. Default is OFF; the default name is "vsl.txt"	*Log ON* *Log ON Filename* *Log OFF*
ABOUT	Print program info. Prints some program information (release, date, recognized commands) to the Stdout and/or Log File	*About*

Tab. 7.4: Commands tab

An example of a .vsl script is shown in **Fig. 7.11**.

```
################################################################################
# VME Scritping Language: Sample
################################################################################

# Enable echo to Stdout and LogFile
Echo ON
Log ON ./Sample_log.txt

# Print the Program Release
About

# Some definitions for the VME access
Alias D16 2      # D16 access
Alias D32 4      # D32 access
Alias A32 0x09   # A32 User Data

# Definition of variables
Var i
Var a
Var stat

# Definition of the board register addresses
Alias CTRL   0x0002
Alias STATUS 0x0080

# Set the Base Address of the board
BaseAddress = 0xEE000000

# Read Status Register
Read D16 A32 STATUS
If (Result != 0) Goto VME_Error
stat = RData
stat &= 0x0040
If (stat == 0)  Print "Status OK"
Else        Print "Status is not OK"

# Loop
i = 0
Repeat
    If (i == 2)  Print "index i is now 2"
    a = 20
    a *= i
    Write D16 A32 CTRL a
    i += 1
Until (i < 4)

Print "End of Loop. %d iterations executed" i
Print "Press a key..."
Goto Exit

VME_Error:
Print "Cannot Access the board at BaseAddress = %08X" BaseAddress

Exit:
Pause
```

Fig. 7.11: Example of vsl script

8 Technical Support

To contact CAEN specialists for requests on the software, hardware, and board return and repair, it is necessary a MyCAEN+ account on www.caen.it:

https://www.caen.it/support-services/getting-started-with-mycaen-portal/

All the instructions for use the Support platform are in the document:

A paper copy of the document is delivered with CAEN boards.

The document is downloadable for free in PDF digital format at:

https://www.caen.it/wp-content/uploads/2022/11/Safety_information_Product_support_W.pdf

CAENVMELib Library User Manual
Interface library for CAEN VME Bridges

CAEN GmbH
Brunnenweg 9
64331 Weiterstadt
Phone +49 (0)212 254 4077
Mobile +49 (0)151 16 548 484
info@caen-de.com
www.caen-de.com

CAEN Technologies, Inc.
1 Edgewater Street - Suite 101
Staten Island, NY 10305
USA
Phone: +1 (718) 981-0401
Fax: +1 (718) 556-9185
info@caentechnologies.com
www.caentechnologies.com

CAEN S.p.A.
Via Vetraia 11
55049 - Viareggio
Italy
Phone +39 0584 388 398
Fax +39 0584 388 959
info@caen.it
www.caen.it

CAENspa INDIA Private Limited
B205, BLDG42, B Wing,
Azad Nagar Sangam CHS,
Mhada Layout, Azad Nagar, Andheri (W)
Mumbai, Mumbai City,
Maharashtra, India, 400053
info@caen-india.in
www.caen-india.in

UM7715 - CAENVMELib Library rev. 6 - June 27th, 2024 00101/18:VMELib.MUTX/03