

## 2015 Test beam Run Control

Generated by Doxygen 1.8.9.1

Mon May 4 2015 19:16:44



# Contents

<b>1</b>	<b>Module Index</b>	<b>1</b>
1.1	Modules . . . . .	1
<b>2</b>	<b>Namespace Index</b>	<b>3</b>
2.1	Namespace List . . . . .	3
<b>3</b>	<b>Hierarchical Index</b>	<b>5</b>
3.1	Class Hierarchy . . . . .	5
<b>4</b>	<b>Data Structure Index</b>	<b>7</b>
4.1	Data Structures . . . . .	7
<b>5</b>	<b>Module Documentation</b>	<b>9</b>
5.1	Socket communication objects . . . . .	9
5.1.1	Detailed Description . . . . .	9
5.1.2	Enumeration Type Documentation . . . . .	9
5.1.2.1	SocketType . . . . .	9
<b>6</b>	<b>Namespace Documentation</b>	<b>11</b>
6.1	VME Namespace Reference . . . . .	11
6.1.1	Typedef Documentation . . . . .	12
6.1.1.1	TDCEventCollection . . . . .	12
6.1.2	Enumeration Type Documentation . . . . .	12
6.1.2.1	acq_mode . . . . .	12
6.1.2.2	ctl_reg . . . . .	12
6.1.2.3	det_mode . . . . .	12
6.1.2.4	micro_handshake . . . . .	13
6.1.2.5	mod_reg . . . . .	13
6.1.2.6	stat_reg . . . . .	14
6.1.2.7	trailead_edge_lsb . . . . .	14
6.1.2.8	trig_conf . . . . .	14
6.2	VME::TDCV1x90Opcodes Namespace Reference . . . . .	14
6.2.1	Function Documentation . . . . .	16

6.2.1.1	AUTOLOAD_DEF_CONFI . . . . .	16
6.2.1.2	AUTOLOAD_USER_CONF . . . . .	16
6.2.1.3	CLEAR_KEEP_TOKEN . . . . .	16
6.2.1.4	CONT_STOR . . . . .	16
6.2.1.5	DIS_ALL_CHANNEL . . . . .	16
6.2.1.6	DIS_CHANNEL . . . . .	16
6.2.1.7	DIS_ERROR_BYPASS . . . . .	16
6.2.1.8	DIS_ERROR_MARK . . . . .	16
6.2.1.9	DIS_HEAD_TRAILER . . . . .	16
6.2.1.10	DIS_SUB_TRG . . . . .	16
6.2.1.11	EN_ALL_CHANNEL . . . . .	16
6.2.1.12	EN_CHANNEL . . . . .	16
6.2.1.13	EN_ERROR_BYPASS . . . . .	16
6.2.1.14	EN_ERROR_MARK . . . . .	16
6.2.1.15	EN_HEAD_TRAILER . . . . .	16
6.2.1.16	EN_SUB_TRG . . . . .	16
6.2.1.17	LOAD_DEF_CONFIG . . . . .	16
6.2.1.18	LOAD_USER_CONFIG . . . . .	16
6.2.1.19	READ_ACQ_MOD . . . . .	16
6.2.1.20	READ_DEAD_TIME . . . . .	16
6.2.1.21	READ_DETECTION . . . . .	16
6.2.1.22	READ_EN_PATTERN . . . . .	16
6.2.1.23	READ_EN_PATTERN32 . . . . .	16
6.2.1.24	READ_ERROR_TYPES . . . . .	16
6.2.1.25	READ_EVENT_SIZE . . . . .	16
6.2.1.26	READ_FIFO_SIZE . . . . .	16
6.2.1.27	READ_GLOB_OFFS . . . . .	16
6.2.1.28	READ_HEAD_TRAILER . . . . .	17
6.2.1.29	READ_RC_ADJ . . . . .	17
6.2.1.30	READ_RES . . . . .	17
6.2.1.31	READ_TRG_CONF . . . . .	17
6.2.1.32	SAVE_RC_ADJ . . . . .	17
6.2.1.33	SAVE_USER_CONFIG . . . . .	17
6.2.1.34	SET_DEAD_TIME . . . . .	17
6.2.1.35	SET_DETECTION . . . . .	17
6.2.1.36	SET_ERROR_TYPES . . . . .	17
6.2.1.37	SET_EVENT_SIZE . . . . .	17
6.2.1.38	SET_FIFO_SIZE . . . . .	17
6.2.1.39	SET_GLOB_OFFS . . . . .	17
6.2.1.40	SET_KEEP_TOKEN . . . . .	17

6.2.1.41	SET_PAIR_RES	17
6.2.1.42	SET_RC_ADJ	17
6.2.1.43	SET_REJ_MARGIN	17
6.2.1.44	SET_SW_MARGIN	17
6.2.1.45	SET_TR_LEAD_LSB	17
6.2.1.46	SET_WIN_OFFS	17
6.2.1.47	SET_WIN_WIDTH	17
6.2.1.48	TRG_MATCH	17
6.2.1.49	WRITE_EN_PATTERN	17
6.2.1.50	WRITE_EN_PATTERN32	17
<b>7</b>	<b>Data Structure Documentation</b>	<b>19</b>
7.1	VME::BridgeVx718 Class Reference	19
7.1.1	Detailed Description	19
7.1.2	Constructor & Destructor Documentation	20
7.1.2.1	BridgeVx718	20
7.1.2.2	~BridgeVx718	20
7.1.3	Member Function Documentation	20
7.1.3.1	GetHandle	20
7.1.3.2	InputConf	20
7.1.3.3	InputRead	20
7.1.3.4	OutputConf	20
7.1.3.5	OutputOff	20
7.1.3.6	OutputOn	20
7.1.4	Field Documentation	20
7.1.4.1	fHandle	21
7.1.4.2	fPortMapping	21
7.2	Client Class Reference	21
7.2.1	Detailed Description	22
7.2.2	Constructor & Destructor Documentation	22
7.2.2.1	Client	22
7.2.2.2	Client	22
7.2.2.3	~Client	23
7.2.3	Member Function Documentation	23
7.2.3.1	Announce	23
7.2.3.2	Connect	23
7.2.3.3	Disconnect	24
7.2.3.4	GetType	24
7.2.3.5	ParseMessage	24
7.2.3.6	Receive	25

7.2.3.7	Send	25
7.2.4	Field Documentation	25
7.2.4.1	fClientId	25
7.2.4.2	flsConnected	25
7.3	Exception Class Reference	25
7.3.1	Detailed Description	26
7.3.2	Constructor & Destructor Documentation	26
7.3.2.1	Exception	26
7.3.2.2	Exception	26
7.3.2.3	~Exception	26
7.3.3	Member Function Documentation	27
7.3.3.1	Description	27
7.3.3.2	Dump	27
7.3.3.3	ErrorNumber	27
7.3.3.4	From	27
7.3.3.5	Type	27
7.3.3.6	TypeString	27
7.3.4	Field Documentation	27
7.3.4.1	fDescription	27
7.3.4.2	fErrorNumber	27
7.3.4.3	fFrom	27
7.3.4.4	fType	28
7.4	file_header_t Struct Reference	28
7.4.1	Detailed Description	28
7.4.2	Field Documentation	28
7.4.2.1	magic	28
7.4.2.2	num_hptdc	28
7.4.2.3	run_id	28
7.4.2.4	spill_id	28
7.5	FileReader Class Reference	28
7.5.1	Constructor & Destructor Documentation	29
7.5.1.1	FileReader	29
7.5.1.2	~FileReader	29
7.5.2	Member Function Documentation	29
7.5.2.1	GetNextEvent	29
7.5.2.2	GetNumTDCs	29
7.5.3	Field Documentation	29
7.5.3.1	fFile	29
7.5.3.2	fHeader	29
7.6	VME::glob_offs Struct Reference	30

7.6.1	Field Documentation	30
7.6.1.1	coarse	30
7.6.1.2	fine	30
7.7	HTTPMessage Class Reference	30
7.7.1	Detailed Description	31
7.7.2	Constructor & Destructor Documentation	31
7.7.2.1	HTTPMessage	32
7.7.2.2	HTTPMessage	32
7.7.3	Member Function Documentation	32
7.7.3.1	Decode	32
7.7.3.2	Dump	32
7.7.3.3	Encode	32
7.7.3.4	GetKey	32
7.7.4	Field Documentation	32
7.7.4.1	fOriginalString	32
7.7.4.2	fWS	32
7.8	Message Class Reference	33
7.8.1	Detailed Description	33
7.8.2	Constructor & Destructor Documentation	34
7.8.2.1	Message	34
7.8.2.2	Message	34
7.8.2.3	Message	34
7.8.2.4	~Message	34
7.8.3	Member Function Documentation	34
7.8.3.1	Dump	34
7.8.3.2	GetKey	34
7.8.3.3	GetString	34
7.8.3.4	IsFromWeb	34
7.8.4	Field Documentation	34
7.8.4.1	fString	34
7.9	Messenger Class Reference	35
7.9.1	Detailed Description	36
7.9.2	Constructor & Destructor Documentation	36
7.9.2.1	Messenger	36
7.9.2.2	Messenger	36
7.9.2.3	~Messenger	37
7.9.3	Member Function Documentation	37
7.9.3.1	AddClient	37
7.9.3.2	Broadcast	37
7.9.3.3	Connect	38

7.9.3.4	Disconnect	38
7.9.3.5	DisconnectClient	39
7.9.3.6	GetType	39
7.9.3.7	ProcessMessage	39
7.9.3.8	Receive	40
7.9.3.9	Send	40
7.9.3.10	SwitchClientType	41
7.9.4	Field Documentation	41
7.9.4.1	fNumAttempts	41
7.9.4.2	fWS	41
7.10	Socket Class Reference	42
7.10.1	Detailed Description	43
7.10.2	Member Typedef Documentation	44
7.10.2.1	SocketCollection	44
7.10.3	Constructor & Destructor Documentation	44
7.10.3.1	Socket	44
7.10.3.2	Socket	44
7.10.3.3	~Socket	44
7.10.4	Member Function Documentation	44
7.10.4.1	AcceptConnections	44
7.10.4.2	Bind	44
7.10.4.3	Configure	45
7.10.4.4	Create	45
7.10.4.5	DumpConnected	45
7.10.4.6	FetchMessage	45
7.10.4.7	GetPort	45
7.10.4.8	GetSocketId	45
7.10.4.9	GetSocketType	45
7.10.4.10	IsWebSocket	45
7.10.4.11	Listen	45
7.10.4.12	PrepareConnection	46
7.10.4.13	SelectConnections	46
7.10.4.14	SendMessage	46
7.10.4.15	SetPort	46
7.10.4.16	SetSocketId	46
7.10.4.17	Start	47
7.10.4.18	Stop	47
7.10.5	Field Documentation	47
7.10.5.1	fAddress	47
7.10.5.2	fBuffer	47



7.10.5.3	fMaster	47
7.10.5.4	fPort	47
7.10.5.5	fReadFds	47
7.10.5.6	fSocketId	47
7.10.5.7	fSocketsConnected	47
7.11	SocketMessage Class Reference	48
7.11.1	Detailed Description	49
7.11.2	Constructor & Destructor Documentation	50
7.11.2.1	SocketMessage	50
7.11.2.2	SocketMessage	50
7.11.2.3	SocketMessage	50
7.11.2.4	SocketMessage	50
7.11.2.5	SocketMessage	50
7.11.2.6	SocketMessage	51
7.11.2.7	SocketMessage	51
7.11.2.8	SocketMessage	51
7.11.2.9	SocketMessage	51
7.11.2.10	SocketMessage	52
7.11.2.11	SocketMessage	52
7.11.2.12	~SocketMessage	52
7.11.3	Member Function Documentation	52
7.11.3.1	Dump	52
7.11.3.2	GetIntValue	52
7.11.3.3	GetKey	53
7.11.3.4	GetString	53
7.11.3.5	GetValue	53
7.11.3.6	GetVectorValue	53
7.11.3.7	Object	53
7.11.3.8	SetKeyValue	53
7.11.3.9	SetKeyValue	53
7.11.3.10	SetKeyValue	54
7.11.3.11	SetKeyValue	54
7.11.3.12	String	54
7.11.4	Field Documentation	54
7.11.4.1	fMessage	54
7.12	VME::TDCEvent Class Reference	54
7.12.1	Detailed Description	55
7.12.2	Member Enumeration Documentation	56
7.12.2.1	EventType	56
7.12.3	Constructor & Destructor Documentation	56

7.12.3.1	TDCEvent	56
7.12.3.2	TDCEvent	56
7.12.3.3	~TDCEvent	56
7.12.4	Member Function Documentation	56
7.12.4.1	GetBunchId	56
7.12.4.2	GetChannelId	57
7.12.4.3	GetErrorFlags	57
7.12.4.4	GetETTT	57
7.12.4.5	GetEventCount	57
7.12.4.6	GetEventId	58
7.12.4.7	GetGeo	58
7.12.4.8	GetLeadingTime	58
7.12.4.9	GetTDCId	59
7.12.4.10	GetTrailingTime	59
7.12.4.11	GetType	59
7.12.4.12	GetWidth	59
7.12.4.13	GetWordCount	60
7.12.4.14	IsTrailing	60
7.12.4.15	SetWord	60
7.12.5	Field Documentation	60
7.12.5.1	fWord	60
7.13	VME::TDCV1x90 Class Reference	61
7.13.1	Detailed Description	62
7.13.2	Constructor & Destructor Documentation	63
7.13.2.1	TDCV1x90	63
7.13.2.2	~TDCV1x90	63
7.13.3	Member Function Documentation	63
7.13.3.1	abort	63
7.13.3.2	CheckConfiguration	64
7.13.3.3	GetBLTEventNumberRegister	64
7.13.3.4	GetCtlRegister	64
7.13.3.5	GetETTT	65
7.13.3.6	GetEventCounter	65
7.13.3.7	GetEvents	65
7.13.3.8	GetEventStored	65
7.13.3.9	GetFirmwareRev	66
7.13.3.10	GetModel	66
7.13.3.11	GetOUI	66
7.13.3.12	GetSerialNumber	67
7.13.3.13	GetStatusRegister	67

7.13.3.14 GetTDCEncapsulation . . . . .	67
7.13.3.15 HardwareReset . . . . .	67
7.13.3.16 IsTriggerMatching . . . . .	68
7.13.3.17 ReadDetection . . . . .	68
7.13.3.18 ReadFIFOSize . . . . .	69
7.13.3.19 ReadGlobalOffset . . . . .	69
7.13.3.20 ReadRCAdjust . . . . .	70
7.13.3.21 ReadRegister . . . . .	70
7.13.3.22 ReadRegister . . . . .	70
7.13.3.23 ReadResolution . . . . .	71
7.13.3.24 ReadTrigConf . . . . .	71
7.13.3.25 SetAcquisitionMode . . . . .	71
7.13.3.26 SetBLTEventNumberRegister . . . . .	72
7.13.3.27 SetContinuousStorage . . . . .	72
7.13.3.28 SetCtlRegister . . . . .	72
7.13.3.29 SetDetection . . . . .	73
7.13.3.30 SetETTT . . . . .	73
7.13.3.31 SetFIFOSize . . . . .	73
7.13.3.32 SetGlobalOffset . . . . .	74
7.13.3.33 SetLSBTraileadEdge . . . . .	74
7.13.3.34 SetPairModeResolution . . . . .	74
7.13.3.35 SetPol . . . . .	75
7.13.3.36 SetRCAdjust . . . . .	75
7.13.3.37 SetStatusRegister . . . . .	75
7.13.3.38 SetTDCEncapsulation . . . . .	75
7.13.3.39 SetTDCErrorMarks . . . . .	76
7.13.3.40 SetTriggerMatching . . . . .	76
7.13.3.41 SetVerboseLevel . . . . .	76
7.13.3.42 SetWindowOffset . . . . .	77
7.13.3.43 SetWindowWidth . . . . .	77
7.13.3.44 SoftwareClear . . . . .	77
7.13.3.45 SoftwareReset . . . . .	78
7.13.3.46 WaitMicro . . . . .	78
7.13.3.47 WriteRegister . . . . .	78
7.13.3.48 WriteRegister . . . . .	78
7.13.4 Field Documentation . . . . .	78
7.13.4.1 acqm . . . . .	78
7.13.4.2 am . . . . .	79
7.13.4.3 am_blt . . . . .	79
7.13.4.4 detm . . . . .	79

7.13.4.5	fBaseAddr	79
7.13.4.6	fBuffer	79
7.13.4.7	fDetMode	79
7.13.4.8	fHandle	79
7.13.4.9	fVerb	79
7.13.4.10	gEnd	79
7.13.4.11	nchannels	79
7.13.4.12	outBufTDCErr	79
7.13.4.13	outBufTDCHHeadTrail	79
7.13.4.14	outBufTDCTTT	79
7.13.4.15	pair_lead_res	79
7.13.4.16	pair_width_res	79
7.13.4.17	trailead_edge_res	79
7.14	VME::trailead_t Struct Reference	79
7.14.1	Field Documentation	79
7.14.1.1	ettt	79
7.14.1.2	event_count	80
7.14.1.3	leading	80
7.14.1.4	total_hits	80
7.14.1.5	trailing	80
<b>Index</b>		<b>81</b>

# Chapter 1

## Module Index

### 1.1 Modules

Here is a list of all modules:

Socket communication objects . . . . .	9
--	---



## Chapter 2

# Namespace Index

### 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">VME</a> . . . . .	<a href="#">11</a>
<a href="#">VME::TDCV1x90Opcodes</a> . . . . .	<a href="#">14</a>





## Chapter 3

# Hierarchical Index

### 3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

VME::BridgeVx718 . . . . .	19
Exception . . . . .	25
file_header_t . . . . .	28
FileReader . . . . .	28
VME::glob_offs . . . . .	30
Message . . . . .	33
HTTPMessage . . . . .	30
SocketMessage . . . . .	48
Socket . . . . .	42
Client . . . . .	21
Messenger . . . . .	35
VME::TDCEvent . . . . .	54
VME::TDCV1x90 . . . . .	61
VME::trailead_t . . . . .	79



## Chapter 4

# Data Structure Index

### 4.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">VME::BridgeVx718</a>	
Class defining the <a href="#">VME</a> bridge . . . . .	19
<a href="#">Client</a>	
Base client object for the socket . . . . .	21
<a href="#">Exception</a>	
A simple exception handler . . . . .	25
<a href="#">file_header_t</a>	
Header to the output files . . . . .	28
<a href="#">FileReader</a> . . . . .	28
<a href="#">VME::glob_offs</a> . . . . .	30
<a href="#">HTTPMessage</a>	
<a href="#">Message</a> to be transmitted through a WebSocket protocol . . . . .	30
<a href="#">Message</a>	
Base socket message type . . . . .	33
<a href="#">Messenger</a>	
Base master object for the socket . . . . .	35
<a href="#">Socket</a>	
Base socket object from which clients/master from a socket inherit . . . . .	42
<a href="#">SocketMessage</a>	
Socket-passed message type . . . . .	48
<a href="#">VME::TDCEvent</a>	
HPTDC event parser . . . . .	54
<a href="#">VME::TDCV1x90</a> . . . . .	61
<a href="#">VME::trailead_t</a> . . . . .	79



## Chapter 5

# Module Documentation

### 5.1 Socket communication objects

#### Data Structures

- class [Client](#)  
*Base client object for the socket.*
- class [HTTPMessage](#)  
*Message to be transmitted through a WebSocket protocol.*
- class [Messenger](#)  
*Base master object for the socket.*
- class [Socket](#)  
*Base socket object from which clients/master from a socket inherit.*
- class [SocketMessage](#)  
*Socket-passed message type.*

#### Enumerations

- enum [Socket::SocketType](#) {  
[Socket::INVALID](#) = -1, [Socket::MASTER](#) = 0, [Socket::WEBSOCKET\\_CLIENT](#), [Socket::CLIENT](#),  
[Socket::DETECTOR](#) }  
*Type of actor playing a role on the socket.*

#### 5.1.1 Detailed Description

#### 5.1.2 Enumeration Type Documentation

##### 5.1.2.1 enum [Socket::SocketType](#)

Type of actor playing a role on the socket.

Enumerator

***INVALID***  
***MASTER***  
***WEBSOCKET\_CLIENT***  
***CLIENT***  
***DETECTOR***



## Chapter 6

# Namespace Documentation

### 6.1 VME Namespace Reference

#### Namespaces

- [TDCV1x90Opcodes](#)

#### Data Structures

- class [BridgeVx718](#)  
*class defining the VME bridge*
- struct [glob\\_offs](#)
- class [TDCEvent](#)  
*HPTDC event parser.*
- class [TDCV1x90](#)
- struct [trailead\\_t](#)

#### Typedefs

- typedef std::vector< [TDCEvent](#) > [TDCEventCollection](#)

#### Enumerations

- enum [trig\\_conf](#) {  
    [MATCH\\_WIN\\_WIDTH](#) = 0, [WIN\\_OFFSET](#) = 1, [EXTRA\\_SEARCH\\_WIN\\_WIDTH](#) = 2, [REJECT\\_MARGIN](#) = 3,  
    [TRIG\\_TIME\\_SUB](#) = 4 }
- enum [trailead\\_edge\\_lsb](#) { [r800ps](#) = 0, [r200ps](#) = 1, [r100ps](#) = 2, [r25ps](#) = 3 }
- enum [micro\\_handshake](#) { [WRITE\\_OK](#) = 0, [READ\\_OK](#) = 1 }
- enum [acq\\_mode](#) { [CONT\\_STORAGE](#), [TRIG\\_MATCH](#) }
- enum [det\\_mode](#) { [PAIR](#) = 0, [OTRILING](#) = 1, [OLEADING](#) = 2, [TRAILEAD](#) = 3 }
- enum [stat\\_reg](#) {  
    [DATA\\_READY](#) = 0, [ALM\\_FULL](#) = 1, [FULL](#) = 2, [TRG\\_MATCH](#) = 3,  
    [HEADER\\_EN](#) = 4, [TERM\\_ON](#) = 5, [ERROR0](#) = 6, [ERROR1](#) = 7,  
    [ERROR2](#) = 8, [ERROR3](#) = 9, [BERR\\_FLAG](#) = 10, [PURG](#) = 11,  
    [RES\\_1](#) = 12, [RES\\_2](#) = 13, [PAIRED](#) = 14, [TRIGGER\\_LOST](#) = 15 }
- enum [ctl\\_reg](#) {  
    [BERREN](#) = 0, [TERM](#) = 1, [TERM\\_SW](#) = 2, [EMPTY\\_EVENT](#) = 3,  
    [ALIGN64](#) = 4, [COMPENSATION\\_ENABLE](#) = 5, [TEST\\_FIFO\\_ENABLE](#) = 6, [READ\\_COMPENSATION\\_SR](#) = 7,  
    [EVENT\\_FIFO\\_ENABLE](#) = 8, [EXTENDED\\_TRIGGER\\_TIME\\_TAG\\_ENABLE](#) = 9 }

- enum `mod_reg` {  
`Control` = 0x1000, `Status` = 0x1002, `InterruptLevel` = 0x100a, `InterruptVector` = 0x100c,  
`GeoAddress` = 0x100e, `MCSTBase` = 0x1010, `MCSTControl` = 0x1012, `ModuleReset` = 0x1014,  
`kSoftwareClear` = 0x1016, `EventCounter` = 0x101c, `EventStored` = 0x1020, `BLTEventNumber` = 0x1024,  
`FirmwareRev` = 0x1026, `Micro` = 0x102e, `MicroHandshake` = 0x1030, `EventFIFO` = 0x1038,  
`EventFIFOStoredRegister` = 0x103c, `EventFIFOStatusRegister` = 0x103e, `ROMOui2` = 0x4024, `ROMOui1` =  
0x4028,  
`ROMOui0` = 0x402c, `ROMBoard2` = 0x4034, `ROMBoard1` = 0x4038, `ROMBoard0` = 0x403c,  
`ROMRevis3` = 0x4040, `ROMRevis2` = 0x4044, `ROMRevis1` = 0x4048, `ROMRevis0` = 0x404c,  
`ROMSerNum1` = 0x4080, `ROMSerNum0` = 0x4084 }

### 6.1.1 Typedef Documentation

6.1.1.1 typedef std::vector<TDCEvent> `VME::TDCEventCollection`

### 6.1.2 Enumeration Type Documentation

6.1.2.1 enum `VME::acq_mode`

Enumerator

***CONT\_STORAGE***

***TRIG\_MATCH***

6.1.2.2 enum `VME::ctl_reg`

Enumerator

***BERREN***

***TERM***

***TERM\_SW***

***EMPTY\_EVENT***

***ALIGN64***

***COMPENSATION\_ENABLE***

***TEST\_FIFO\_ENABLE***

***READ\_COMPENSATION\_SRAM\_ENABLE***

***EVENT\_FIFO\_ENABLE***

***EXTENDED\_TRIGGER\_TIME\_TAG\_ENABLE***

6.1.2.3 enum `VME::det_mode`

Enumerator

***PAIR***

***OTRAILING***

***OLEADING***

***TRAILEAD***



## 6.1.2.4 enum VME::micro\_handshake

Enumerator

**WRITE\_OK** Is the TDC ready for writing?**READ\_OK** Is the TDC ready for reading?

## 6.1.2.5 enum VME::mod\_reg

Enumerator

**Control****Status****InterruptLevel****InterruptVector****GeoAddress****MCSTBase****MCSTControl****ModuleReset****kSoftwareClear****EventCounter****EventStored****BLTEventNumber****FirmwareRev****Micro****MicroHandshake****EventFIFO****EventFIFOStoredRegister****EventFIFOStatusRegister****ROMQui2****ROMQui1****ROMQui0****ROMBoard2****ROMBoard1****ROMBoard0****ROMRevis3****ROMRevis2****ROMRevis1****ROMRevis0****ROMSerNum1****ROMSerNum0**

## 6.1.2.6 enum VME::stat\_reg

Enumerator

***DATA\_READY***  
***ALM\_FULL***  
***FULL***  
***TRG\_MATCH***  
***HEADER\_EN***  
***TERM\_ON***  
***ERROR0***  
***ERROR1***  
***ERROR2***  
***ERROR3***  
***BERR\_FLAG***  
***PURG***  
***RES\_1***  
***RES\_2***  
***PAIRED***  
***TRIGGER\_LOST***

## 6.1.2.7 enum VME::trailead\_edge\_lsb

Enumerator

***r800ps***  
***r200ps***  
***r100ps***  
***r25ps***

## 6.1.2.8 enum VME::trig\_conf

Enumerator

***MATCH\_WIN\_WIDTH***  
***WIN\_OFFSET***  
***EXTRA\_SEARCH\_WIN\_WIDTH***  
***REJECT\_MARGIN***  
***TRIG\_TIME\_SUB***

## 6.2 VME::TDCV1x90Opcodes Namespace Reference

Functions

- Opcode [TRG\\_MATCH](#) (0x0000)
- Opcode [CONT\\_STOR](#) (0x0100)
- Opcode [READ\\_ACQ\\_MOD](#) (0x0200)
- Opcode [SET\\_KEEP\\_TOKEN](#) (0x0300)

- Opcode [CLEAR\\_KEEP\\_TOKEN](#) (0x0400)
- Opcode [LOAD\\_DEF\\_CONFIG](#) (0x0500)
- Opcode [SAVE\\_USER\\_CONFIG](#) (0x0600)
- Opcode [LOAD\\_USER\\_CONFIG](#) (0x0700)
- Opcode [AUTOLOAD\\_USER\\_CONF](#) (0x0800)
- Opcode [AUTOLOAD\\_DEF\\_CONFI](#) (0x0900)
- Opcode [SET\\_WIN\\_WIDTH](#) (0x1000)
- Opcode [SET\\_WIN\\_OFFS](#) (0x1100)
- Opcode [SET\\_SW\\_MARGIN](#) (0x1200)
- Opcode [SET\\_REJ\\_MARGIN](#) (0x1300)
- Opcode [EN\\_SUB\\_TRG](#) (0x1400)
- Opcode [DIS\\_SUB\\_TRG](#) (0x1500)
- Opcode [READ\\_TRG\\_CONF](#) (0x1600)
- Opcode [SET\\_DETECTION](#) (0x2200)
- Opcode [READ\\_DETECTION](#) (0x2300)
- Opcode [SET\\_TR\\_LEAD\\_LSB](#) (0x2400)
- Opcode [SET\\_PAIR\\_RES](#) (0x2500)
- Opcode [READ\\_RES](#) (0x2600)
- Opcode [SET\\_DEAD\\_TIME](#) (0x2800)
- Opcode [READ\\_DEAD\\_TIME](#) (0x2900)
- Opcode [EN\\_HEAD\\_TRAILER](#) (0x3000)
- Opcode [DIS\\_HEAD\\_TRAILER](#) (0x3100)
- Opcode [READ\\_HEAD\\_TRAILER](#) (0x3200)
- Opcode [SET\\_EVENT\\_SIZE](#) (0x3300)
- Opcode [READ\\_EVENT\\_SIZE](#) (0x3400)
- Opcode [EN\\_ERROR\\_MARK](#) (0x3500)
- Opcode [DIS\\_ERROR\\_MARK](#) (0x3600)
- Opcode [EN\\_ERROR\\_BYPASS](#) (0x3700)
- Opcode [DIS\\_ERROR\\_BYPASS](#) (0x3800)
- Opcode [SET\\_ERROR\\_TYPES](#) (0x3900)
- Opcode [READ\\_ERROR\\_TYPES](#) (0x3a00)
- Opcode [SET\\_FIFO\\_SIZE](#) (0x3b00)
- Opcode [READ\\_FIFO\\_SIZE](#) (0x3c00)
- Opcode [EN\\_CHANNEL](#) (0x4000)
- Opcode [DIS\\_CHANNEL](#) (0x4100)
- Opcode [EN\\_ALL\\_CHANNEL](#) (0x4200)
- Opcode [DIS\\_ALL\\_CHANNEL](#) (0x4300)
- Opcode [WRITE\\_EN\\_PATTERN](#) (0x4400)
- Opcode [READ\\_EN\\_PATTERN](#) (0x4500)
- Opcode [WRITE\\_EN\\_PATTERN32](#) (0x4600)
- Opcode [READ\\_EN\\_PATTERN32](#) (0x4700)
- Opcode [SET\\_GLOB\\_OFFS](#) (0x5000)
- Opcode [READ\\_GLOB\\_OFFS](#) (0x5100)
- Opcode [SET\\_RC\\_ADJ](#) (0x5400)
- Opcode [READ\\_RC\\_ADJ](#) (0x5500)
- Opcode [SAVE\\_RC\\_ADJ](#) (0x5600)

## 6.2.1 Function Documentation

- 6.2.1.1 Opcode VME::TDCV1x90Opcodes::AUTOLOAD\_DEF\_CONFI ( 0x0900 )
- 6.2.1.2 Opcode VME::TDCV1x90Opcodes::AUTOLOAD\_USER\_CONF ( 0x0800 )
- 6.2.1.3 Opcode VME::TDCV1x90Opcodes::CLEAR\_KEEP\_TOKEN ( 0x0400 )
- 6.2.1.4 Opcode VME::TDCV1x90Opcodes::CONT\_STOR ( 0x0100 )
- 6.2.1.5 Opcode VME::TDCV1x90Opcodes::DIS\_ALL\_CHANNEL ( 0x4300 )
- 6.2.1.6 Opcode VME::TDCV1x90Opcodes::DIS\_CHANNEL ( 0x4100 )
- 6.2.1.7 Opcode VME::TDCV1x90Opcodes::DIS\_ERROR\_BYPASS ( 0x3800 )
- 6.2.1.8 Opcode VME::TDCV1x90Opcodes::DIS\_ERROR\_MARK ( 0x3600 )
- 6.2.1.9 Opcode VME::TDCV1x90Opcodes::DIS\_HEAD\_TRAILER ( 0x3100 )
- 6.2.1.10 Opcode VME::TDCV1x90Opcodes::DIS\_SUB\_TRG ( 0x1500 )
- 6.2.1.11 Opcode VME::TDCV1x90Opcodes::EN\_ALL\_CHANNEL ( 0x4200 )
- 6.2.1.12 Opcode VME::TDCV1x90Opcodes::EN\_CHANNEL ( 0x4000 )
- 6.2.1.13 Opcode VME::TDCV1x90Opcodes::EN\_ERROR\_BYPASS ( 0x3700 )
- 6.2.1.14 Opcode VME::TDCV1x90Opcodes::EN\_ERROR\_MARK ( 0x3500 )
- 6.2.1.15 Opcode VME::TDCV1x90Opcodes::EN\_HEAD\_TRAILER ( 0x3000 )
- 6.2.1.16 Opcode VME::TDCV1x90Opcodes::EN\_SUB\_TRG ( 0x1400 )
- 6.2.1.17 Opcode VME::TDCV1x90Opcodes::LOAD\_DEF\_CONFIG ( 0x0500 )
- 6.2.1.18 Opcode VME::TDCV1x90Opcodes::LOAD\_USER\_CONFIG ( 0x0700 )
- 6.2.1.19 Opcode VME::TDCV1x90Opcodes::READ\_ACQ\_MOD ( 0x0200 )
- 6.2.1.20 Opcode VME::TDCV1x90Opcodes::READ\_DEAD\_TIME ( 0x2900 )
- 6.2.1.21 Opcode VME::TDCV1x90Opcodes::READ\_DETECTION ( 0x2300 )
- 6.2.1.22 Opcode VME::TDCV1x90Opcodes::READ\_EN\_PATTERN ( 0x4500 )
- 6.2.1.23 Opcode VME::TDCV1x90Opcodes::READ\_EN\_PATTERN32 ( 0x4700 )
- 6.2.1.24 Opcode VME::TDCV1x90Opcodes::READ\_ERROR\_TYPES ( 0x3a00 )
- 6.2.1.25 Opcode VME::TDCV1x90Opcodes::READ\_EVENT\_SIZE ( 0x3400 )
- 6.2.1.26 Opcode VME::TDCV1x90Opcodes::READ\_FIFO\_SIZE ( 0x3c00 )
- 6.2.1.27 Opcode VME::TDCV1x90Opcodes::READ\_GLOB\_OFFS ( 0x5100 )

- 6.2.1.28 Opcode VME::TDCV1x90Opcodes::READ\_HEAD\_TRAILER ( 0x3200 )
- 6.2.1.29 Opcode VME::TDCV1x90Opcodes::READ\_RC\_ADJ ( 0x5500 )
- 6.2.1.30 Opcode VME::TDCV1x90Opcodes::READ\_RES ( 0x2600 )
- 6.2.1.31 Opcode VME::TDCV1x90Opcodes::READ\_TRG\_CONF ( 0x1600 )
- 6.2.1.32 Opcode VME::TDCV1x90Opcodes::SAVE\_RC\_ADJ ( 0x5600 )
- 6.2.1.33 Opcode VME::TDCV1x90Opcodes::SAVE\_USER\_CONFIG ( 0x0600 )
- 6.2.1.34 Opcode VME::TDCV1x90Opcodes::SET\_DEAD\_TIME ( 0x2800 )
- 6.2.1.35 Opcode VME::TDCV1x90Opcodes::SET\_DETECTION ( 0x2200 )
- 6.2.1.36 Opcode VME::TDCV1x90Opcodes::SET\_ERROR\_TYPES ( 0x3900 )
- 6.2.1.37 Opcode VME::TDCV1x90Opcodes::SET\_EVENT\_SIZE ( 0x3300 )
- 6.2.1.38 Opcode VME::TDCV1x90Opcodes::SET\_FIFO\_SIZE ( 0x3b00 )
- 6.2.1.39 Opcode VME::TDCV1x90Opcodes::SET\_GLOB\_OFFS ( 0x5000 )
- 6.2.1.40 Opcode VME::TDCV1x90Opcodes::SET\_KEEP\_TOKEN ( 0x0300 )
- 6.2.1.41 Opcode VME::TDCV1x90Opcodes::SET\_PAIR\_RES ( 0x2500 )
- 6.2.1.42 Opcode VME::TDCV1x90Opcodes::SET\_RC\_ADJ ( 0x5400 )
- 6.2.1.43 Opcode VME::TDCV1x90Opcodes::SET\_REJ\_MARGIN ( 0x1300 )
- 6.2.1.44 Opcode VME::TDCV1x90Opcodes::SET\_SW\_MARGIN ( 0x1200 )
- 6.2.1.45 Opcode VME::TDCV1x90Opcodes::SET\_TR\_LEAD\_LSB ( 0x2400 )
- 6.2.1.46 Opcode VME::TDCV1x90Opcodes::SET\_WIN\_OFFS ( 0x1100 )
- 6.2.1.47 Opcode VME::TDCV1x90Opcodes::SET\_WIN\_WIDTH ( 0x1000 )
- 6.2.1.48 Opcode VME::TDCV1x90Opcodes::TRG\_MATCH ( 0x0000 )
- 6.2.1.49 Opcode VME::TDCV1x90Opcodes::WRITE\_EN\_PATTERN ( 0x4400 )
- 6.2.1.50 Opcode VME::TDCV1x90Opcodes::WRITE\_EN\_PATTERN32 ( 0x4600 )



## Chapter 7

# Data Structure Documentation

### 7.1 VME::BridgeVx718 Class Reference

class defining the VME bridge

```
#include <VME_BridgeVx718.h>
```

#### Public Member Functions

- [BridgeVx718](#) (const char \*device, unsigned int type)  
*Constructor.*
- [~BridgeVx718](#) ()  
*Destructor.*
- int32\_t [GetHandle](#) () const  
*Gets bhandle.*
- void [OutputConf](#) (CVOutputSelect output)  
*Set and control the output lines.*
- void [OutputOn](#) (CVOutputSelect output)
- void [OutputOff](#) (CVOutputSelect output)
- void [InputConf](#) (CVInputSelect input)  
*Set and read the input lines.*
- void [InputRead](#) (CVInputSelect input)

#### Private Attributes

- std::map< CVOutputSelect, CVOutputRegisterBits > [fPortMapping](#)  
*Map output lines [0,4] to corresponding register.*
- int32\_t [fHandle](#)  
*Device handle.*

#### 7.1.1 Detailed Description

class defining the VME bridge

This class initializes the CAEN V1718 VME bridge in order to control the crate.

**Author**

Laurent Forthomme [laurent.forthomme@cern.ch](mailto:laurent.forthomme@cern.ch)  
 Bob Velghe [bob.velghe@cern.ch](mailto:bob.velghe@cern.ch)

**Date**

Jun 2010

**7.1.2 Constructor & Destructor Documentation****7.1.2.1 VME::BridgeVx718::BridgeVx718 ( const char \* *device*, unsigned int *type* )**

Constructor.

Bridge class constructor

**Parameters**

in	<i>device</i>	Device identifier on the <a href="#">VME</a> crate
in	<i>type</i>	Device type (1718/2718)

**7.1.2.2 VME::BridgeVx718::~~BridgeVx718 ( )**

Destructor.

Bridge class destructor

**7.1.3 Member Function Documentation****7.1.3.1 int32\_t VME::BridgeVx718::GetHandle ( ) const [inline]**

Gets bhandle.

Gives bhandle value

**Returns**

bhandle value

**7.1.3.2 void VME::BridgeVx718::InputConf ( CVInputSelect *input* )**

Set and read the input lines.

**7.1.3.3 void VME::BridgeVx718::InputRead ( CVInputSelect *input* )****7.1.3.4 void VME::BridgeVx718::OutputConf ( CVOutputSelect *output* )**

Set and control the output lines.

**7.1.3.5 void VME::BridgeVx718::OutputOff ( CVOutputSelect *output* )****7.1.3.6 void VME::BridgeVx718::OutputOn ( CVOutputSelect *output* )****7.1.4 Field Documentation**



7.1.4.1 `int32_t VME::BridgeVx718::fHandle` `[private]`

Device handle.

7.1.4.2 `std::map<CVOutputSelect,CVOutputRegisterBits> VME::BridgeVx718::fPortMapping` `[private]`

Map output lines [0,4] to corresponding register.

The documentation for this class was generated from the following files:

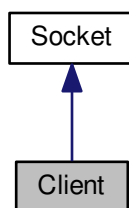
- `include/VME_BridgeVx718.h`
- `src/VME_BridgeVx718.cpp`

## 7.2 Client Class Reference

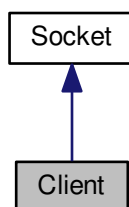
Base client object for the socket.

```
#include <Client.h>
```

Inheritance diagram for Client:



Collaboration diagram for Client:



### Public Member Functions

- [Client](#) ()

*General void client constructor.*

- [Client](#) (int port)

*Bind a socket client to a given port.*

- virtual [~Client](#) ()
- bool [Connect](#) ()

*Bind this client to the socket.*

- void [Disconnect](#) ()

*Unbind this client from the socket.*

- void [Send](#) (const [Message](#) &m) const

*Send a message to the master through the socket.*

- void [Receive](#) ()

*Receive a socket message from the master.*

- virtual void [ParseMessage](#) (const [SocketMessage](#) &m)

*Parse a [SocketMessage](#) received from the master.*

- virtual [SocketType](#) [GetType](#) () const

*[Socket](#) actor type retrieval method.*

## Private Member Functions

- void [Announce](#) ()

*Announce our entry on the socket to its master.*

## Private Attributes

- int [fClientId](#)
- bool [flsConnected](#)

## Additional Inherited Members

### 7.2.1 Detailed Description

Base client object for the socket.

[Client](#) object used by the server to send/receive commands from the messenger/broadcaster.

#### Author

Laurent Forthomme [laurent.forthomme@cern.ch](mailto:laurent.forthomme@cern.ch)

#### Date

24 Mar 2015

### 7.2.2 Constructor & Destructor Documentation

#### 7.2.2.1 [Client::Client](#) ( ) [inline]

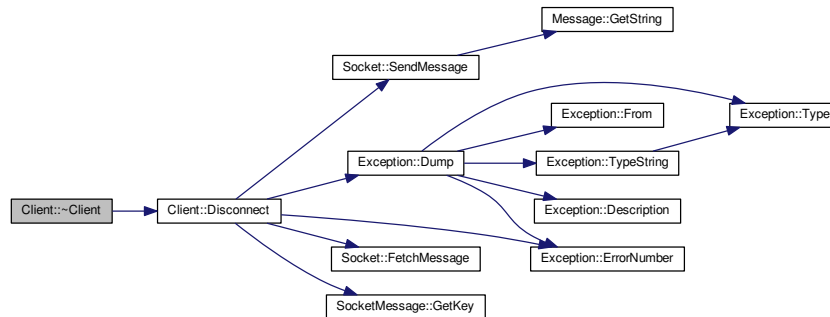
General void client constructor.

#### 7.2.2.2 [Client::Client](#) ( int *port* )

Bind a socket client to a given port.

## 7.2.2.3 Client::~~Client( ) [virtual]

Here is the call graph for this function:

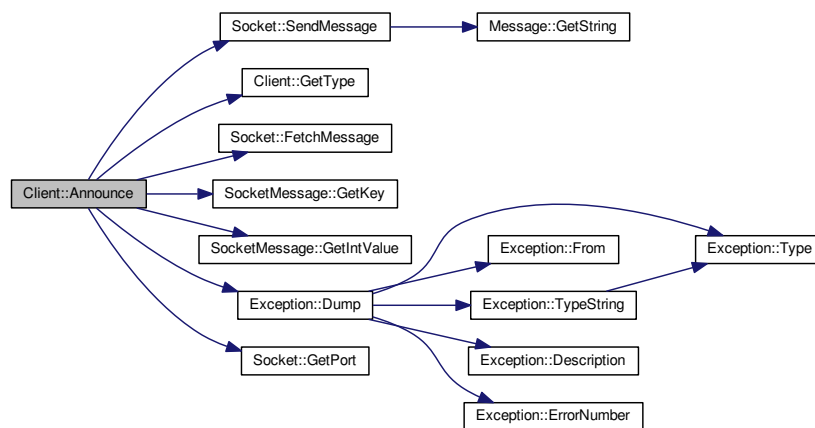


## 7.2.3 Member Function Documentation

## 7.2.3.1 void Client::Announce( ) [private]

Announce our entry on the socket to its master.

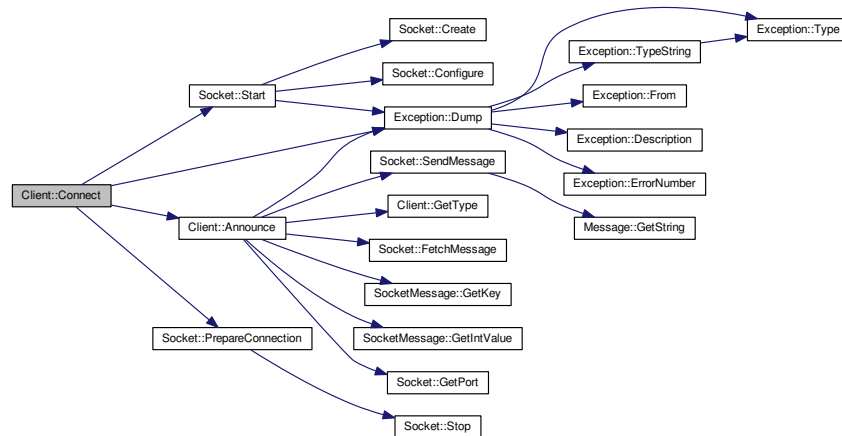
Here is the call graph for this function:



## 7.2.3.2 bool Client::Connect( )

Bind this client to the socket.

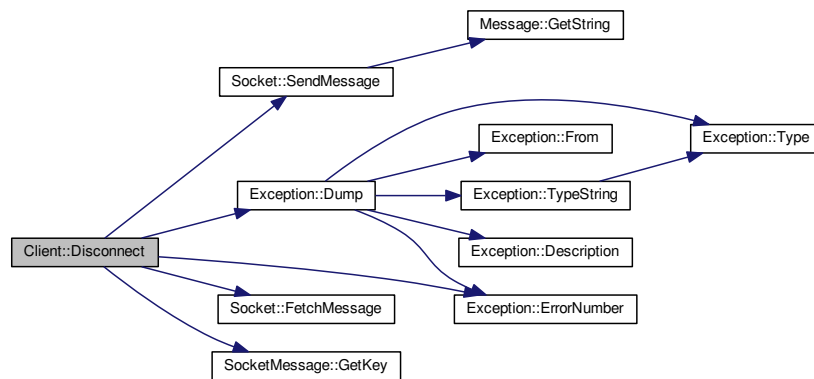
Here is the call graph for this function:



#### 7.2.3.3 void Client::Disconnect ( )

Unbind this client from the socket.

Here is the call graph for this function:



#### 7.2.3.4 virtual SocketType Client::GetType ( ) const [inline],[virtual]

[Socket](#) actor type retrieval method.

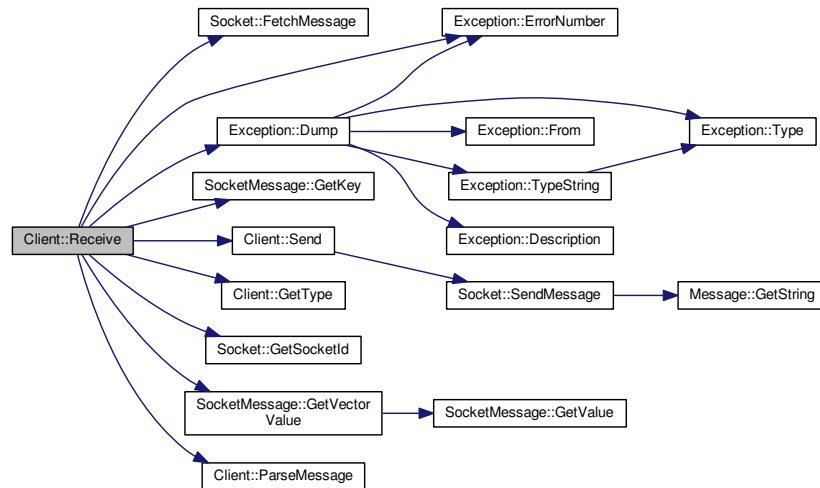
#### 7.2.3.5 virtual void Client::ParseMessage ( const SocketMessage & m ) [inline],[virtual]

Parse a [SocketMessage](#) received from the master.

## 7.2.3.6 void Client::Receive ( )

Receive a socket message from the master.

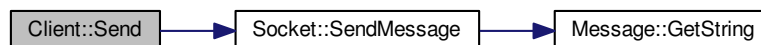
Here is the call graph for this function:



## 7.2.3.7 void Client::Send ( const Message &amp; m ) const [inline]

Send a message to the master through the socket.

Here is the call graph for this function:



## 7.2.4 Field Documentation

## 7.2.4.1 int Client::fClientId [private]

## 7.2.4.2 bool Client::fIsConnected [private]

The documentation for this class was generated from the following files:

- include/Client.h
- src/Client.cpp

## 7.3 Exception Class Reference

A simple exception handler.

```
#include <Exception.h>
```

## Public Member Functions

- [Exception](#) (const char \*from, std::string desc, ExceptionType type=Undefined, const int id=0)
- [Exception](#) (const char \*from, const char \*desc, ExceptionType type=Undefined, const int id=0)
- [~Exception](#) ()
- std::string [From](#) () const
- int [ErrorNumber](#) () const
- std::string [Description](#) () const
- ExceptionType [Type](#) () const
- std::string [TypeString](#) () const
- void [Dump](#) (std::ostream &os=std::cerr) const

## Private Attributes

- std::string [fFrom](#)
- std::string [fDescription](#)
- ExceptionType [fType](#)
- int [fErrorNumber](#)

### 7.3.1 Detailed Description

A simple exception handler.

#### Author

Laurent Forthomme [laurent.forthomme@cern.ch](mailto:laurent.forthomme@cern.ch)

#### Date

24 Mar 2015

### 7.3.2 Constructor & Destructor Documentation

**7.3.2.1** `Exception::Exception ( const char * from, std::string desc, ExceptionType type = Undefined, const int id = 0 )`  
[inline]

**7.3.2.2** `Exception::Exception ( const char * from, const char * desc, ExceptionType type = Undefined, const int id = 0 )`  
[inline]

**7.3.2.3** `Exception::~~Exception ( )` [inline]

Here is the call graph for this function:

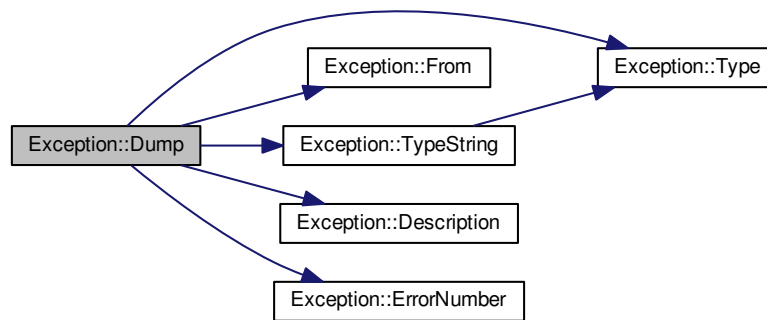


### 7.3.3 Member Function Documentation

7.3.3.1 `std::string Exception::Description ( ) const` [inline]

7.3.3.2 `void Exception::Dump ( std::ostream & os = std::cerr ) const` [inline]

Here is the call graph for this function:



7.3.3.3 `int Exception::ErrorNumber ( ) const` [inline]

7.3.3.4 `std::string Exception::From ( ) const` [inline]

7.3.3.5 `ExceptionType Exception::Type ( ) const` [inline]

7.3.3.6 `std::string Exception::TypeString ( ) const` [inline]

Here is the call graph for this function:



### 7.3.4 Field Documentation

7.3.4.1 `std::string Exception::fDescription` [private]

7.3.4.2 `int Exception::fErrorNumber` [private]

7.3.4.3 `std::string Exception::fFrom` [private]

#### 7.3.4.4 ExceptionType Exception::fType [private]

The documentation for this class was generated from the following file:

- include/Exception.h

## 7.4 file\_header\_t Struct Reference

Header to the output files.

```
#include <FileConstants.h>
```

### Data Fields

- uint32\_t [magic](#)
- uint32\_t [run\\_id](#)
- uint32\_t [spill\\_id](#)
- uint8\_t [num\\_hptdc](#)

#### 7.4.1 Detailed Description

Header to the output files.

General header to store in each collected data file for offline readout. It enable any reader to retrieve the run/spill number, as well as the HPTDC configuration during data collection.

#### Author

Laurent Forthomme [laurent.forthomme@cern.ch](mailto:laurent.forthomme@cern.ch)

#### Date

14 Apr 2015

#### 7.4.2 Field Documentation

##### 7.4.2.1 uint32\_t file\_header\_t::magic

##### 7.4.2.2 uint8\_t file\_header\_t::num\_hptdc

##### 7.4.2.3 uint32\_t file\_header\_t::run\_id

##### 7.4.2.4 uint32\_t file\_header\_t::spill\_id

The documentation for this struct was generated from the following file:

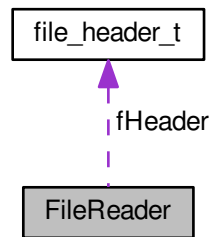
- include/FileConstants.h

## 7.5 FileReader Class Reference

```
#include <FileReader.h>
```



Collaboration diagram for FileReader:



### Public Member Functions

- [FileReader](#) (std::string name)
- [~FileReader](#) ()
- unsigned int [GetNumTDCs](#) () const
- [VME::TDCEvent](#) [GetNextEvent](#) ()

### Private Attributes

- std::ifstream [fFile](#)
- [file\\_header\\_t](#) [fHeader](#)

### 7.5.1 Constructor & Destructor Documentation

7.5.1.1 `FileReader::FileReader ( std::string name )`

7.5.1.2 `FileReader::~~FileReader ( )`

### 7.5.2 Member Function Documentation

7.5.2.1 `VME::TDCEvent FileReader::GetNextEvent ( )` `[inline]`

7.5.2.2 `unsigned int FileReader::GetNumTDCs ( ) const` `[inline]`

### 7.5.3 Field Documentation

7.5.3.1 `std::ifstream FileReader::fFile` `[private]`

7.5.3.2 `file_header_t FileReader::fHeader` `[private]`

The documentation for this class was generated from the following files:

- include/FileReader.h
- src/FileReader.cpp

## 7.6 VME::glob\_offs Struct Reference

```
#include <VME_TDCV1x90.h>
```

### Data Fields

- uint16\_t [coarse](#)
- uint16\_t [fine](#)

### 7.6.1 Field Documentation

7.6.1.1 uint16\_t VME::glob\_offs::coarse

7.6.1.2 uint16\_t VME::glob\_offs::fine

The documentation for this struct was generated from the following file:

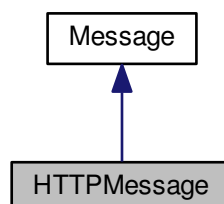
- include/VME\_TDCV1x90.h

## 7.7 HTTPMessage Class Reference

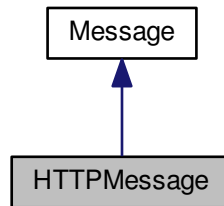
[Message](#) to be transmitted through a WebSocket protocol.

```
#include <HTTPMessage.h>
```

Inheritance diagram for HTTPMessage:



Collaboration diagram for HTTPMessage:



### Public Member Functions

- [HTTPMessage](#) (WebSocket \*ws, [Message](#) m, MessageAction a)
- [HTTPMessage](#) (WebSocket \*ws, const char \*msg, MessageAction a)
- void [Decode](#) ()
- void [Encode](#) ()
- MessageKey [GetKey](#) () const
- void [Dump](#) (std::ostream &os=std::cout) const

### Private Attributes

- WebSocket \* [fWS](#)
- std::string [fOriginalString](#)

### Additional Inherited Members

#### 7.7.1 Detailed Description

[Message](#) to be transmitted through a WebSocket protocol.

Type of message compatible to the transmission through a WebSocket protocol. It enables a direct conversion of standards from any socket message format used elsewhere in this code using the *MessageAction* statement.

#### Author

Laurent Forthomme [laurent.forthomme@cern.ch](mailto:laurent.forthomme@cern.ch)

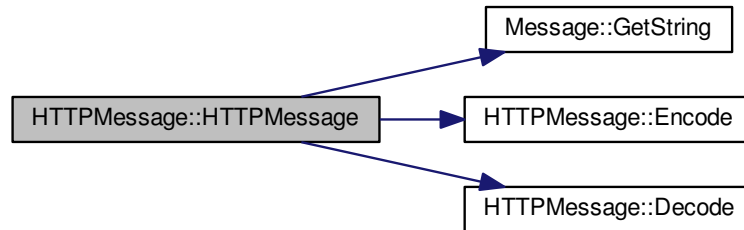
#### Date

1 Apr 2015

#### 7.7.2 Constructor & Destructor Documentation

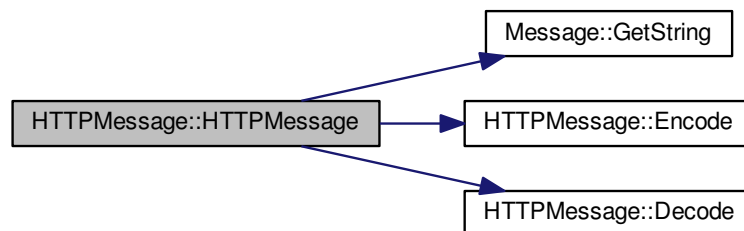
#### 7.7.2.1 HTTPMessage::HTTPMessage ( WebSocket \* *ws*, Message *m*, MessageAction *a* ) [inline]

Here is the call graph for this function:



#### 7.7.2.2 HTTPMessage::HTTPMessage ( WebSocket \* *ws*, const char \* *msg*, MessageAction *a* ) [inline]

Here is the call graph for this function:



### 7.7.3 Member Function Documentation

#### 7.7.3.1 void HTTPMessage::Decode ( ) [inline]

#### 7.7.3.2 void HTTPMessage::Dump ( std::ostream & *os* = std::cout ) const [inline]

#### 7.7.3.3 void HTTPMessage::Encode ( ) [inline]

#### 7.7.3.4 MessageKey HTTPMessage::GetKey ( ) const [inline]

### 7.7.4 Field Documentation

#### 7.7.4.1 std::string HTTPMessage::fOriginalString [private]

#### 7.7.4.2 WebSocket\* HTTPMessage::fWS [private]

The documentation for this class was generated from the following file:

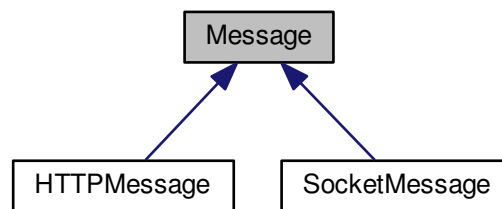
- `include/HTTPMessage.h`

## 7.8 Message Class Reference

Base socket message type.

```
#include <Message.h>
```

Inheritance diagram for Message:



### Public Member Functions

- `Message ()`  
*Void message constructor.*
- `Message (const char *msg)`  
*Construct a message from a string.*
- `Message (std::string msg)`  
*Construct a message from a string.*
- `virtual ~Message ()`
- `MessageKey GetKey () const`  
*Placeholder for the MessageKey retrieval method.*
- `std::string GetString () const`  
*Retrieve the string carried by this message as a whole.*
- `bool IsFromWeb () const`  
*Extract from any message its potential arrival from a WebSocket protocol.*
- `void Dump (std::ostream &os=std::cout) const`

### Protected Attributes

- `std::string fString`

#### 7.8.1 Detailed Description

Base socket message type.

Base handler for messages to be transmitted through the socket

**Author**

Laurent Forthomme [laurent.forthomme@cern.ch](mailto:laurent.forthomme@cern.ch)

**Date**

6 Apr 2015

**7.8.2 Constructor & Destructor Documentation****7.8.2.1 `Message::Message ( )` [inline]**

Void message constructor.

**7.8.2.2 `Message::Message ( const char * msg )` [inline]**

Construct a message from a string.

**7.8.2.3 `Message::Message ( std::string msg )` [inline]**

Construct a message from a string.

**7.8.2.4 `virtual Message::~Message ( )` [inline],[virtual]****7.8.3 Member Function Documentation****7.8.3.1 `void Message::Dump ( std::ostream & os = std::cout ) const` [inline]****7.8.3.2 `MessageKey Message::GetKey ( ) const` [inline]**

Placeholder for the MessageKey retrieval method.

**7.8.3.3 `std::string Message::GetString ( ) const` [inline]**

Retrieve the string carried by this message as a whole.

**7.8.3.4 `bool Message::IsFromWeb ( ) const` [inline]**

Extract from any message its potential arrival from a WebSocket protocol.

**7.8.4 Field Documentation****7.8.4.1 `std::string Message::fString` [protected]**

The documentation for this class was generated from the following file:

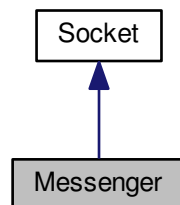
- include/Message.h

## 7.9 Messenger Class Reference

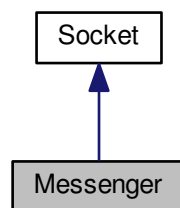
Base master object for the socket.

```
#include <Messenger.h>
```

Inheritance diagram for Messenger:



Collaboration diagram for Messenger:



### Public Member Functions

- [Messenger](#) ()  
*Build a void master object or socket actor.*
- [Messenger](#) (int port)  
*Build a master object to control the socket.*
- [~Messenger](#) ()
- bool [Connect](#) ()  
*Connect the master to the socket.*
- void [Disconnect](#) ()  
*Remove the master and destroy the socket.*
- void [Send](#) (const [Message](#) &m, int sid) const  
*Send any type of message to any client.*
- void [Receive](#) ()  
*Handle a message reception from a client.*
- void [Broadcast](#) (const [Message](#) &m) const

*Emit a message to all clients connected through the socket.*

- `SocketType GetType ()` const  
*Socket actor type retrieval method.*

## Private Member Functions

- void `AddClient ()`  
*Add a client to listen to.*
- void `DisconnectClient` (int sid, MessageKey key, bool force=false)  
*Disconnect a client.*
- void `SwitchClientType` (int sid, `Socket::SocketType` type)
- void `ProcessMessage` (`SocketMessage` m, int sid)  
*Process a message received from the socket.*

## Private Attributes

- `WebSocket` \* `fWS`
- int `fNumAttempts`

## Additional Inherited Members

### 7.9.1 Detailed Description

Base master object for the socket.

Messenger/broadcaster object used by the server to send/receive commands from the clients/listeners.

#### Author

Laurent Forthomme [laurent.forthomme@cern.ch](mailto:laurent.forthomme@cern.ch)

#### Date

23 Mar 2015

### 7.9.2 Constructor & Destructor Documentation

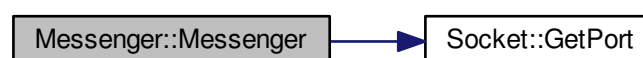
#### 7.9.2.1 `Messenger::Messenger ( )`

Build a void master object or socket actor.

#### 7.9.2.2 `Messenger::Messenger ( int port )`

Build a master object to control the socket.

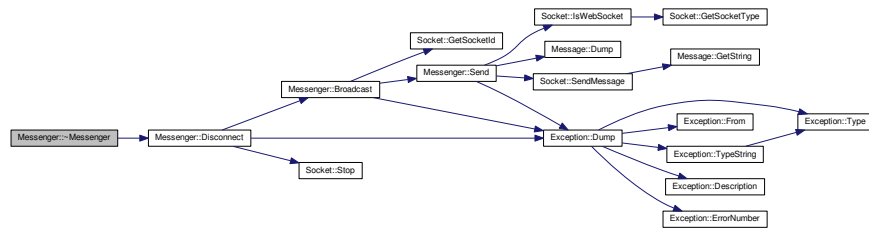
Here is the call graph for this function:





## 7.9.2.3 Messenger::~~Messenger ( )

Here is the call graph for this function:



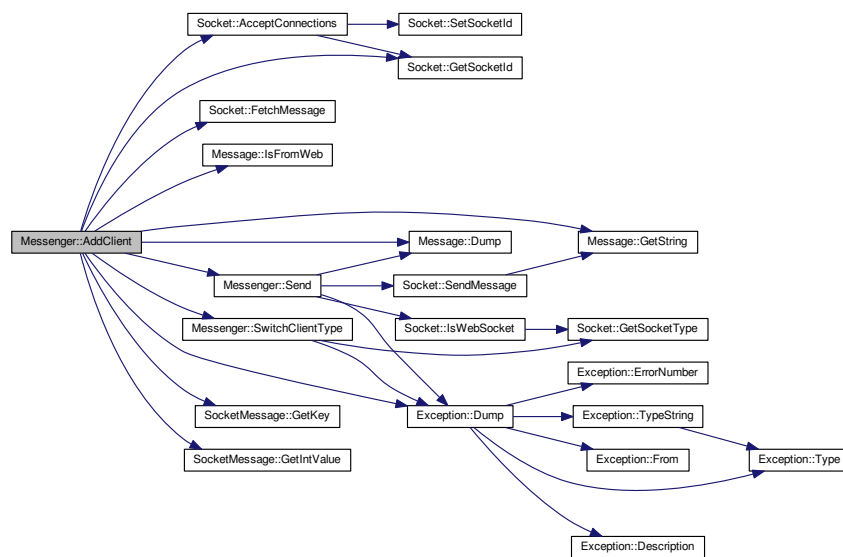
## 7.9.3 Member Function Documentation

## 7.9.3.1 void Messenger::AddClient ( ) [private]

Add a client to listen to.

Add one client to the list of socket actors to monitor for message retrieval/submission.

Here is the call graph for this function:



## 7.9.3.2 void Messenger::Broadcast ( const Message &amp; m ) const

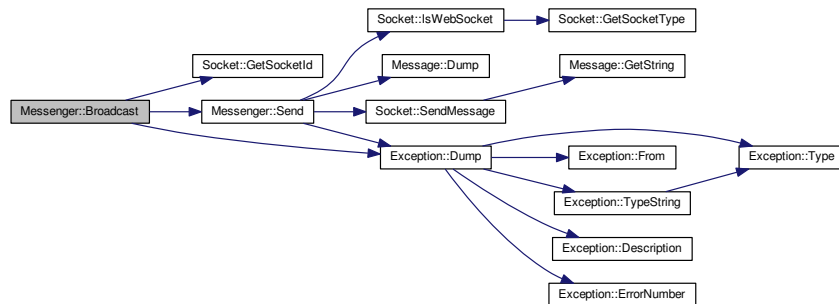
Emit a message to all clients connected through the socket.

Parameters

---

in	m	Message to transmit
----	---	---------------------

Here is the call graph for this function:

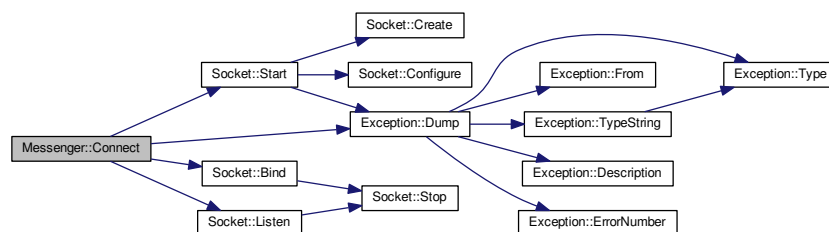


#### 7.9.3.3 bool Messenger::Connect ( )

Connect the master to the socket.

Connect this master to the socket for clients to be able to bind.

Here is the call graph for this function:

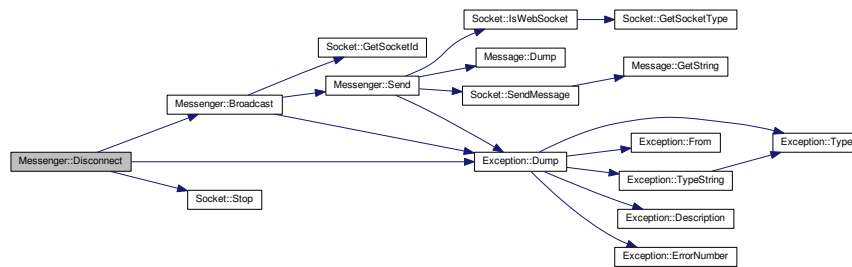


#### 7.9.3.4 void Messenger::Disconnect ( )

Remove the master and destroy the socket.

Remove this master from the socket, thus disconnecting automatically the clients connected.

Here is the call graph for this function:



**7.9.3.5** `void Messenger::DisconnectClient ( int sid, MessageKey key, bool force = false )` [private]

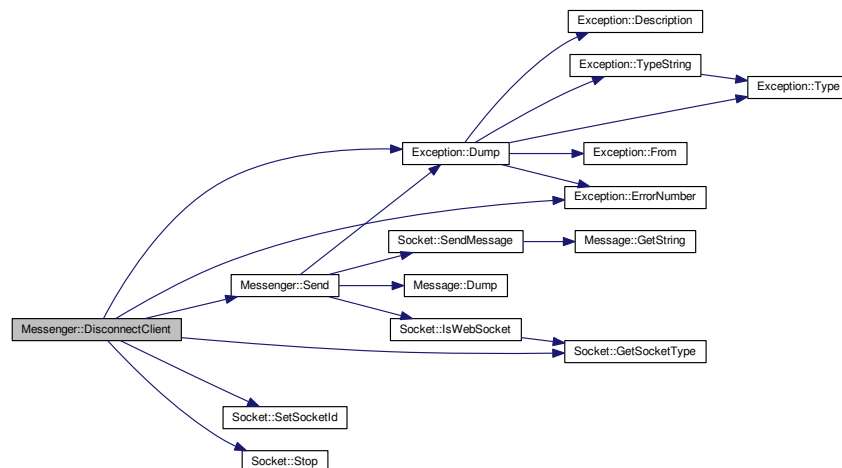
Disconnect a client.

Ask to a client to disconnect from this socket.

#### Parameters

in	<i>sid</i>	Unique identifier of the client to disconnect
in	<i>key</i>	Key to the message to transmit for disconnection
in	<i>force</i>	Do we need to force the client out of this socket ?

Here is the call graph for this function:



**7.9.3.6** `SocketType Messenger::GetType ( ) const` [inline]

[Socket](#) actor type retrieval method.

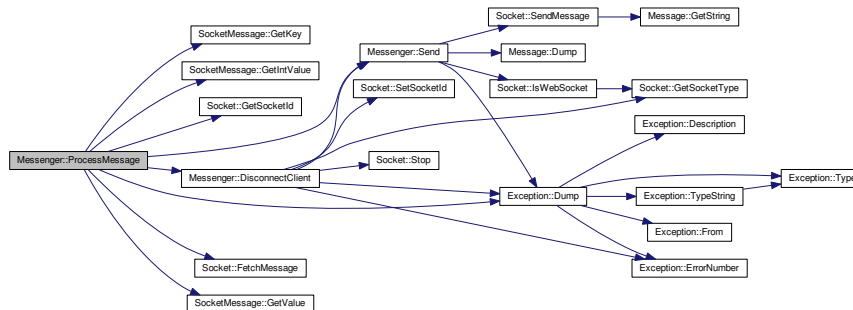
**7.9.3.7** `void Messenger::ProcessMessage ( SocketMessage m, int sid )` [private]

Process a message received from the socket.

## Parameters

in	<i>Unique</i>	identifier of the client sending the message
----	---------------	--

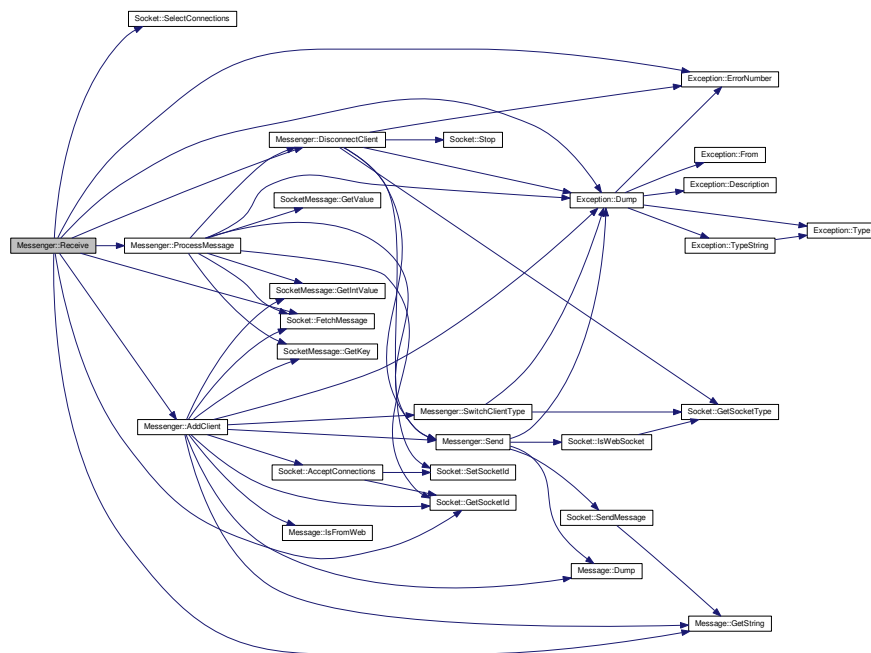
Here is the call graph for this function:



## 7.9.3.8 void Messenger::Receive ( )

Handle a message reception from a client.

Here is the call graph for this function:



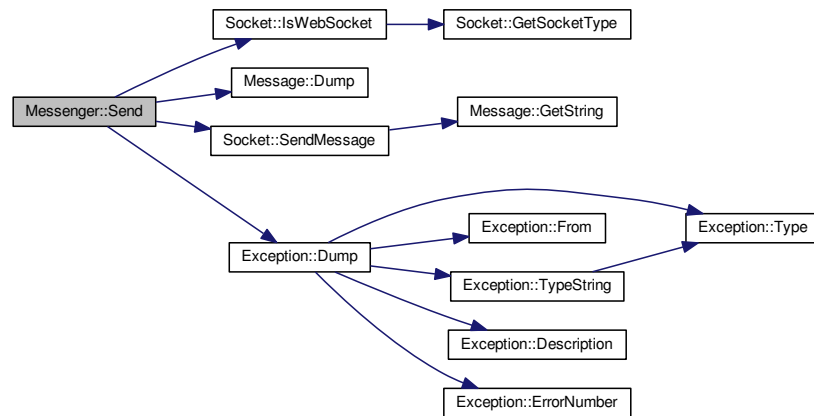
## 7.9.3.9 void Messenger::Send ( const Message &amp; m, int sid ) const [inline]

Send any type of message to any client.

## Parameters

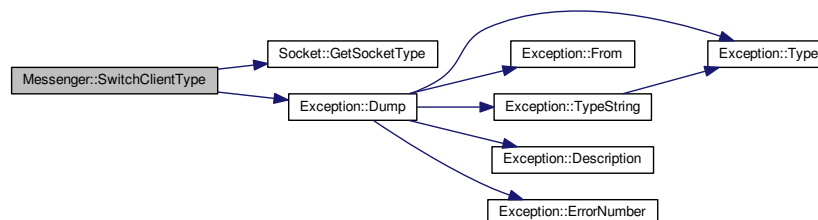
in	<i>m</i>	<a href="#">Message</a> to transmit
in	<i>sid</i>	Unique identifier of the client on this socket

Here is the call graph for this function:



#### 7.9.3.10 void Messenger::SwitchClientType ( int *sid*, Socket::SocketType *type* ) [private]

Here is the call graph for this function:



### 7.9.4 Field Documentation

#### 7.9.4.1 int Messenger::fNumAttempts [private]

#### 7.9.4.2 WebSocket\* Messenger::fWS [private]

The documentation for this class was generated from the following files:

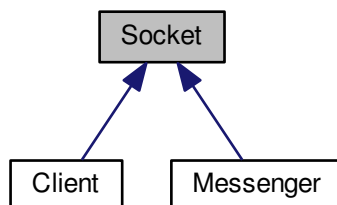
- include/Messenger.h
- src/Messenger.cpp

## 7.10 Socket Class Reference

Base socket object from which clients/master from a socket inherit.

```
#include <Socket.h>
```

Inheritance diagram for Socket:



### Public Types

- enum `SocketType` {  
`INVALID` = -1, `MASTER` = 0, `WEBSOCKET_CLIENT`, `CLIENT`,  
`DETECTOR` }
- *Type of actor playing a role on the socket.*
- typedef std::set< std::pair< int, `SocketType` > > `SocketCollection`

### Public Member Functions

- `Socket` ()
- `Socket` (int port)
- virtual `~Socket` ()
- void `Stop` ()  
*Terminates the socket and all attached communications.*
- void `SetPort` (int port)
- int `GetPort` () const  
*Retrieve the port used for this socket.*
- void `AcceptConnections` (`Socket` &socket)  
*Accept connection from a client.*
- void `SelectConnections` ()
- void `SetSocketId` (int sid)
- int `GetSocketId` () const
- `SocketType` `GetSocketType` (int sid) const
- bool `IsWebSocket` (int sid) const
- void `DumpConnected` () const

### Protected Member Functions

- bool `Start` ()  
*Start the socket.*

- void [Bind](#) ()  
*Bind a name to a socket.*
- void [PrepareConnection](#) ()
- void [Listen](#) (int maxconn)  
*Listen to incoming messages.*
- void [SendMessage](#) ([Message](#) message, int id=-1) const  
*Send a message on a socket.*
- [Message](#) [FetchMessage](#) (int id=-1) const  
*Receive a message from a socket.*

### Protected Attributes

- int [fPort](#)
- char [fBuffer](#) [MAX\_WORD\_LENGTH]
- [SocketCollection](#) [fSocketsConnected](#)
- fd\_set [fMaster](#)  
*Master file descriptor list.*
- fd\_set [fReadFds](#)  
*Temp file descriptor list for select()*

### Private Member Functions

- void [Create](#) ()  
*Create an endpoint for communication.*
- void [Configure](#) ()  
*Configure the socket object for communication.*

### Private Attributes

- int [fSocketId](#)
- struct sockaddr\_in [fAddress](#)

#### 7.10.1 Detailed Description

Base socket object from which clients/master from a socket inherit.

General object providing all useful method to connect/bind/send/receive information through system sockets.

#### Author

Laurent Forthomme [laurent.forthomme@cern.ch](mailto:laurent.forthomme@cern.ch)

#### Date

23 Mar 2015

## 7.10.2 Member Typedef Documentation

7.10.2.1 `typedef std::set< std::pair<int,SocketType> > Socket::SocketCollection`

## 7.10.3 Constructor & Destructor Documentation

7.10.3.1 `Socket::Socket ( ) [inline]`

7.10.3.2 `Socket::Socket ( int port )`

7.10.3.3 `Socket::~~Socket ( ) [virtual]`

## 7.10.4 Member Function Documentation

7.10.4.1 `void Socket::AcceptConnections ( Socket & socket )`

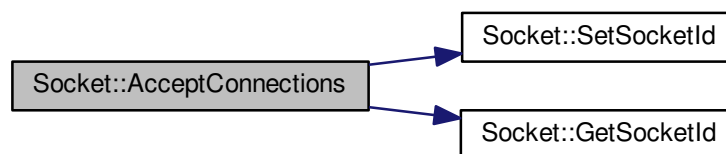
Accept connection from a client.

Set the socket to accept connections any client transmitting through the socket

### Parameters

<code>in, out</code>	<code><i>socket</i></code>	Master/client object to enable on the socket
----------------------	----------------------------	--

Here is the call graph for this function:



7.10.4.2 `void Socket::Bind ( ) [protected]`

Bind a name to a socket.

### Returns

Success of the operation

Here is the call graph for this function:





**7.10.4.3** `void Socket::Configure ( ) [private]`

Configure the socket object for communication.

**7.10.4.4** `void Socket::Create ( ) [private]`

Create an endpoint for communication.

**7.10.4.5** `void Socket::DumpConnected ( ) const`**7.10.4.6** `Message Socket::FetchMessage ( int id = -1 ) const [protected]`

Receive a message from a socket.

**Returns**

Received message as a `std::string`

**7.10.4.7** `int Socket::GetPort ( ) const [inline]`

Retrieve the port used for this socket.

**7.10.4.8** `int Socket::GetSocketId ( ) const [inline]`**7.10.4.9** `SocketType Socket::GetSocketType ( int sid ) const [inline]`**7.10.4.10** `bool Socket::IsWebSocket ( int sid ) const [inline]`

Here is the call graph for this function:

**7.10.4.11** `void Socket::Listen ( int maxconn ) [protected]`

Listen to incoming messages.

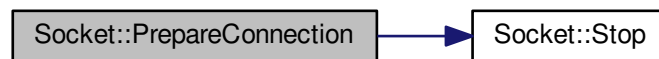
Set the socket to listen to any message coming from outside

Here is the call graph for this function:



#### 7.10.4.12 void Socket::PrepareConnection ( ) [protected]

Here is the call graph for this function:



#### 7.10.4.13 void Socket::SelectConnections ( )

Register all open file descriptors to read their communication through the socket

#### 7.10.4.14 void Socket::SendMessage ( Message message, int id = -1 ) const [protected]

Send a message on a socket.

Here is the call graph for this function:



#### 7.10.4.15 void Socket::SetPort ( int port ) [inline]

#### 7.10.4.16 void Socket::SetSocketId ( int sid ) [inline]

#### 7.10.4.17 `bool Socket::Start ( )` [protected]

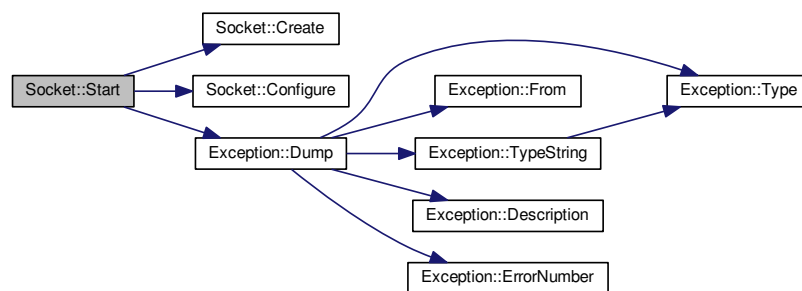
Start the socket.

Launch all mandatory operations to set the socket to be used

##### Returns

Success of the operation

Here is the call graph for this function:



#### 7.10.4.18 `void Socket::Stop ( )`

Terminates the socket and all attached communications.

### 7.10.5 Field Documentation

#### 7.10.5.1 `struct sockaddr_in Socket::fAddress` [private]

#### 7.10.5.2 `char Socket::fBuffer[MAX_WORD_LENGTH]` [protected]

#### 7.10.5.3 `fd_set Socket::fMaster` [protected]

Master file descriptor list.

#### 7.10.5.4 `int Socket::fPort` [protected]

#### 7.10.5.5 `fd_set Socket::fReadFds` [protected]

Temp file descriptor list for select()

#### 7.10.5.6 `int Socket::fSocketId` [private]

A file descriptor for this socket, if *Create* was performed beforehand.

#### 7.10.5.7 `SocketCollection Socket::fSocketsConnected` [protected]

The documentation for this class was generated from the following files:

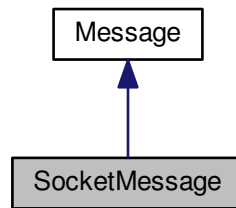
- `include/Socket.h`
- `src/Socket.cpp`

## 7.11 SocketMessage Class Reference

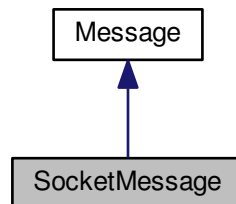
Socket-passed message type.

```
#include <SocketMessage.h>
```

Inheritance diagram for SocketMessage:



Collaboration diagram for SocketMessage:



### Public Member Functions

- [SocketMessage](#) ()
- [SocketMessage](#) (const [Message](#) &msg)
- [SocketMessage](#) (const char \*msg\_s)
- [SocketMessage](#) (std::string msg\_s)
- [SocketMessage](#) (const MessageKey &key)  
*Construct a socket message out of a key.*
- [SocketMessage](#) (const MessageKey &key, const char \*value)  
*Construct a socket message out of a key and a string-type value.*
- [SocketMessage](#) (const MessageKey &key, std::string value)  
*Construct a socket message out of a key and a string-type value.*

- [SocketMessage](#) (const MessageKey &key, const int value)  
*Construct a socket message out of a key and an integer-type value.*
- [SocketMessage](#) (const MessageKey &key, const float value)  
*Construct a socket message out of a key and a float-type value.*
- [SocketMessage](#) (const MessageKey &key, const double value)  
*Construct a socket message out of a key and a double precision-type value.*
- [SocketMessage](#) (MessageMap msg\_m)  
*Construct a socket message out of a map of key/string-type value.*
- [~SocketMessage](#) ()
- void [SetKeyValue](#) (const MessageKey &key, const char \*value)  
*String-valued message.*
- void [SetKeyValue](#) (const MessageKey &key, int int\_value)  
*Send an integer-valued message.*
- void [SetKeyValue](#) (const MessageKey &key, float float\_value)  
*Float-valued message.*
- void [SetKeyValue](#) (const MessageKey &key, double double\_value)  
*Double-valued message.*
- std::string [GetString](#) () const  
*Extract the whole key:value message.*
- MessageKey [GetKey](#) () const  
*Extract the message's key.*
- std::string [GetValue](#) () const  
*Extract the message's string value.*
- int [GetIntValue](#) () const  
*Extract the message's integer value.*
- VectorValue [GetVectorValue](#) () const  
*Extract the message's vector of string value.*
- void [Dump](#) (std::ostream &os=std::cout) const

### Private Member Functions

- MessageMap [Object](#) () const
- std::string [String](#) () const

### Private Attributes

- MessageMap [fMessage](#)

### Additional Inherited Members

#### 7.11.1 Detailed Description

Socket-passed message type.

#### Author

Laurent Forthomme [laurent.forthomme@cern.ch](mailto:laurent.forthomme@cern.ch)

#### Date

26 Mar 2015

## 7.11.2 Constructor & Destructor Documentation

7.11.2.1 `SocketMessage::SocketMessage ( ) [inline]`

7.11.2.2 `SocketMessage::SocketMessage ( const Message & msg ) [inline]`

Here is the call graph for this function:



7.11.2.3 `SocketMessage::SocketMessage ( const char * msg_s ) [inline]`

Here is the call graph for this function:



7.11.2.4 `SocketMessage::SocketMessage ( std::string msg_s ) [inline]`

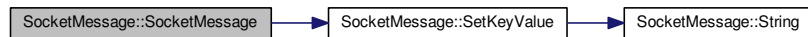
Here is the call graph for this function:



7.11.2.5 `SocketMessage::SocketMessage ( const MessageKey & key ) [inline]`

Construct a socket message out of a key.

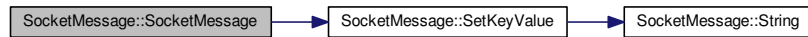
Here is the call graph for this function:



#### 7.11.2.6 SocketMessage::SocketMessage ( const MessageKey & key, const char \* value ) [inline]

Construct a socket message out of a key and a string-type value.

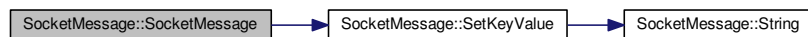
Here is the call graph for this function:



#### 7.11.2.7 SocketMessage::SocketMessage ( const MessageKey & key, std::string value ) [inline]

Construct a socket message out of a key and a string-type value.

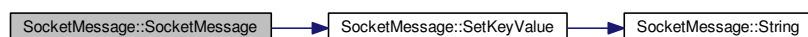
Here is the call graph for this function:



#### 7.11.2.8 SocketMessage::SocketMessage ( const MessageKey & key, const int value ) [inline]

Construct a socket message out of a key and an integer-type value.

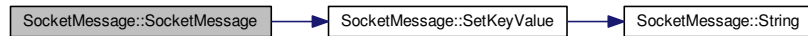
Here is the call graph for this function:



#### 7.11.2.9 SocketMessage::SocketMessage ( const MessageKey & key, const float value ) [inline]

Construct a socket message out of a key and a float-type value.

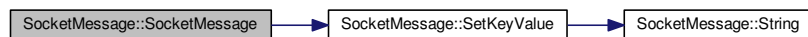
Here is the call graph for this function:



#### 7.11.2.10 `SocketMessage::SocketMessage ( const MessageKey & key, const double value ) [inline]`

Construct a socket message out of a key and a double precision-type value.

Here is the call graph for this function:



#### 7.11.2.11 `SocketMessage::SocketMessage ( MessageMap msg_m ) [inline]`

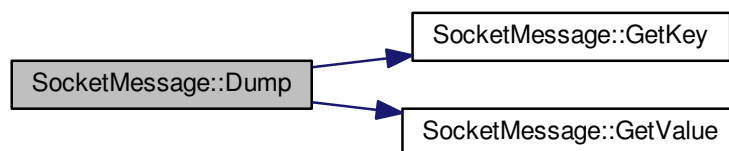
Construct a socket message out of a map of key/string-type value.

#### 7.11.2.12 `SocketMessage::~~SocketMessage ( ) [inline]`

### 7.11.3 Member Function Documentation

#### 7.11.3.1 `void SocketMessage::Dump ( std::ostream & os = std::cout ) const [inline]`

Here is the call graph for this function:



#### 7.11.3.2 `int SocketMessage::GetIntValue ( ) const [inline]`

Extract the message's integer value.



#### 7.11.3.3 MessageKey SocketMessage::GetKey ( ) const [inline]

Extract the message's key.

#### 7.11.3.4 std::string SocketMessage::GetString ( ) const [inline]

Extract the whole key:value message.

#### 7.11.3.5 std::string SocketMessage::GetValue ( ) const [inline]

Extract the message's string value.

#### 7.11.3.6 VectorValue SocketMessage::GetVectorValue ( ) const [inline]

Extract the message's vector of string value.

Here is the call graph for this function:



#### 7.11.3.7 MessageMap SocketMessage::Object ( ) const [inline],[private]

#### 7.11.3.8 void SocketMessage::SetKeyValue ( const MessageKey & key, const char \* value ) [inline]

String-valued message.

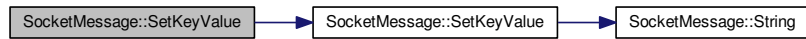
Here is the call graph for this function:



#### 7.11.3.9 void SocketMessage::SetKeyValue ( const MessageKey & key, int int\_value ) [inline]

Send an integer-valued message.

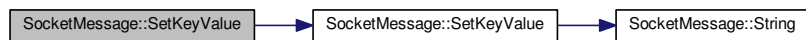
Here is the call graph for this function:



**7.11.3.10** `void SocketMessage::SetKeyValue ( const MessageKey & key, float float_value )` `[inline]`

Float-valued message.

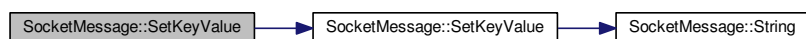
Here is the call graph for this function:



**7.11.3.11** `void SocketMessage::SetKeyValue ( const MessageKey & key, double double_value )` `[inline]`

Double-valued message.

Here is the call graph for this function:



**7.11.3.12** `std::string SocketMessage::String ( ) const` `[inline], [private]`

## 7.11.4 Field Documentation

**7.11.4.1** `MessageMap SocketMessage::fMessage` `[private]`

The documentation for this class was generated from the following file:

- `include/SocketMessage.h`

## 7.12 VME::TDCEvent Class Reference

HPTDC event parser.

```
#include <VME_TDCEvent.h>
```

## Public Types

- enum [EventType](#) {  
[TDCMeasurement](#) = 0x0, [TDCHeader](#) = 0x1, [TDCTrailer](#) = 0x3, [TDCError](#) = 0x4,  
[GlobalHeader](#) = 0x8, [GlobalTrailer](#) = 0x10, [ETTT](#) = 0x11, [Filler](#) = 0x18 }

## Public Member Functions

- [TDCEvent](#) ()
- [TDCEvent](#) (const uint32\_t &word)
- virtual [~TDCEvent](#) ()
- void [SetWord](#) (const uint32\_t &word)
- [EventType](#) [GetType](#) () const  
*Type of packet read out from the TDC.*
- uint8\_t [GetTDCId](#) () const  
*Programmed identifier of master TDC providing the event.*
- uint16\_t [GetEventId](#) () const  
*Event identifier from event counter.*
- uint16\_t [GetWordCount](#) () const  
*Total number of words in event (including headers and trailers)*
- uint8\_t [GetGeo](#) () const
- uint8\_t [GetChannelId](#) () const
- uint32\_t [GetEventCount](#) () const  
*Total number of events.*
- uint16\_t [GetBunchId](#) () const  
*Bunch identifier of trigger (or trigger time tag)*
- bool [IsTrailing](#) () const  
*Are we dealing with a trailing or a leading measurement?*
- uint32\_t [GetETTT](#) () const  
*Extended trigger time tag.*
- uint32\_t [GetLeadingTime](#) (bool pair=false) const  
*Leading edge measurement in programmed time resolution.*
- uint8\_t [GetWidth](#) () const  
*Width of pulse in programmed time resolution.*
- uint32\_t [GetTrailingTime](#) () const  
*Trailing edge measurement in programmed time resolution.*
- uint16\_t [GetErrorFlags](#) () const  
*Return error flags if an error condition has been detected.*

## Private Attributes

- uint32\_t [fWord](#)

### 7.12.1 Detailed Description

HPTDC event parser.

Object enabling to decipher any measurement/error/debug event returned by the HPTDC chip

#### Author

Laurent Forthomme [laurent.forthomme@cern.ch](mailto:laurent.forthomme@cern.ch)

Date

4 May 2015

## 7.12.2 Member Enumeration Documentation

### 7.12.2.1 enum VME::TDCEvent::EventType

Enumerator

***TDCMeasurement***

***TDCHeader***

***TDCTrailer***

***TDCError***

***GlobalHeader***

***GlobalTrailer***

***ETTT***

***Filler***

## 7.12.3 Constructor & Destructor Documentation

7.12.3.1 VME::TDCEvent::TDCEvent ( ) [inline]

7.12.3.2 VME::TDCEvent::TDCEvent ( const uint32\_t & word ) [inline]

7.12.3.3 virtual VME::TDCEvent::~~TDCEvent ( ) [inline],[virtual]

## 7.12.4 Member Function Documentation

7.12.4.1 uint16\_t VME::TDCEvent::GetBunchId ( ) const [inline]

Bunch identifier of trigger (or trigger time tag)

Here is the call graph for this function:



#### 7.12.4.2 uint8\_t VME::TDCEvent::GetChannelId ( ) const [inline]

Here is the call graph for this function:



#### 7.12.4.3 uint16\_t VME::TDCEvent::GetErrorFlags ( ) const [inline]

Return error flags if an error condition has been detected.

Here is the call graph for this function:



#### 7.12.4.4 uint32\_t VME::TDCEvent::GetETTT ( ) const [inline]

Extended trigger time tag.

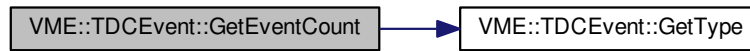
Here is the call graph for this function:



#### 7.12.4.5 uint32\_t VME::TDCEvent::GetEventCount ( ) const [inline]

Total number of events.

Here is the call graph for this function:



#### 7.12.4.6 `uint16_t VME::TDCEvent::GetEventId ( ) const [inline]`

Event identifier from event counter.

Here is the call graph for this function:



#### 7.12.4.7 `uint8_t VME::TDCEvent::GetGeo ( ) const [inline]`

Here is the call graph for this function:



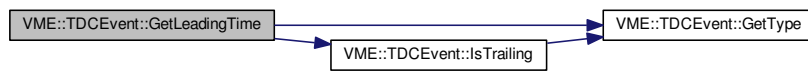
#### 7.12.4.8 `uint32_t VME::TDCEvent::GetLeadingTime ( bool pair = false ) const [inline]`

Leading edge measurement in programmed time resolution.

Parameters

<code>in</code>	<code><i>pair</i></code>	Are we dealing with a pair measurement?
-----------------	--------------------------	---

Here is the call graph for this function:



#### 7.12.4.9 `uint8_t VME::TDCEvent::GetTDCId ( ) const [inline]`

Programmed identifier of master TDC providing the event.

Here is the call graph for this function:



#### 7.12.4.10 `uint32_t VME::TDCEvent::GetTrailingTime ( ) const [inline]`

Trailing edge measurement in programmed time resolution.

Here is the call graph for this function:



#### 7.12.4.11 `EventType VME::TDCEvent::GetType ( ) const [inline]`

Type of packet read out from the TDC.

#### 7.12.4.12 `uint8_t VME::TDCEvent::GetWidth ( ) const [inline]`

Width of pulse in programmed time resolution.

Here is the call graph for this function:



7.12.4.13 `uint16_t VME::TDCEvent::GetWordCount ( ) const [inline]`

Total number of words in event (including headers and trailers)

Here is the call graph for this function:



7.12.4.14 `bool VME::TDCEvent::IsTrailing ( ) const [inline]`

Are we dealing with a trailing or a leading measurement?

Here is the call graph for this function:



7.12.4.15 `void VME::TDCEvent::SetWord ( const uint32_t & word ) [inline]`

## 7.12.5 Field Documentation

7.12.5.1 `uint32_t VME::TDCEvent::fWord [private]`

The documentation for this class was generated from the following file:

- `include/VME_TDCEvent.h`



## 7.13 VME::TDCV1x90 Class Reference

```
#include <VME_TDCV1x90.h>
```

### Public Member Functions

- [TDCV1x90](#) (int32\_t, uint32\_t, [acq\\_mode](#) [acqm](#)=TRIG\_MATCH, [det\\_mode](#) [detm](#)=TRAILEAD)
- [~TDCV1x90](#) ()
- void [SetVerboseLevel](#) (unsigned short [verb](#)=0)
- uint32\_t [GetModel](#) ()
- uint32\_t [GetOUI](#) ()
- uint32\_t [GetSerialNumber](#) ()
- void [CheckConfiguration](#) ()
- void [SetPol](#) (uint16\_t)
- void [SetLSBTraileadEdge](#) ([trailead\\_edge\\_lsb](#))
- void [SetAcquisitionMode](#) ([acq\\_mode](#))
- bool [SetTriggerMatching](#) ()
- bool [IsTriggerMatching](#) ()
- bool [SetContinuousStorage](#) ()
- void [GetFirmwareRev](#) ()
- void [SetGlobalOffset](#) (uint16\_t, uint16\_t)
- [glob\\_offs](#) [ReadGlobalOffset](#) ()
- void [SetRCAdjust](#) (int, uint16\_t)
- uint16\_t [ReadRCAdjust](#) (int)
- uint32\_t [GetEventCounter](#) ()
- uint16\_t [GetEventStored](#) ()
- void [SetDetection](#) ([det\\_mode](#))
- [det\\_mode](#) [ReadDetection](#) ()
- void [SetTDCEncapsulation](#) (bool)
- bool [GetTDCEncapsulation](#) ()
- void [SetTDCErrorMarks](#) (bool)
- void [ReadResolution](#) ([det\\_mode](#))
- void [SetPairModeResolution](#) (int, int)
- void [SetBLTEventNumberRegister](#) (uint16\_t)
- uint16\_t [GetBLTEventNumberRegister](#) ()
- void [SetWindowWidth](#) (uint16\_t)
- void [SetWindowOffset](#) (int16\_t)
- uint16\_t [ReadTrigConf](#) ([trig\\_conf](#))
- bool [WaitMicro](#) ([micro\\_handshake](#))
- bool [SoftwareClear](#) ()
- bool [SoftwareReset](#) ()
- bool [HardwareReset](#) ()
- bool [GetStatusRegister](#) ([stat\\_reg](#))
- void [SetStatusRegister](#) ([stat\\_reg](#), bool)
- bool [GetCtlRegister](#) ([ctl\\_reg](#))
- void [SetCtlRegister](#) ([ctl\\_reg](#), bool)
- void [SetETTT](#) (bool)
- bool [GetETTT](#) ()
- [TDCEventCollection](#) [GetEvents](#) ()
- void [SetFIFOSize](#) (uint16\_t)
- void [ReadFIFOSize](#) ()
- void [abort](#) ()
- void [WriteRegister](#) ([mod\\_reg](#), uint16\_t \*)

*Write on register.*

- void `WriteRegister` (`mod_reg`, `uint32_t *`)  
*Write on register.*
- void `ReadRegister` (`mod_reg`, `uint16_t *`)  
*Read on register.*
- void `ReadRegister` (`mod_reg`, `uint32_t *`)  
*Read on register.*

### Private Attributes

- `uint32_t fBaseAddr`
- `int32_t fHandle`
- `det_mode fDetMode`
- unsigned short `fVerb`
- CVAddressModifier `am`
- CVAddressModifier `am_blt`
- `uint32_t * fBuffer`
- `det_mode detm`
- `acq_mode acqm`
- bool `outBufTDCHHeadTrail`
- bool `outBufTDCErr`
- bool `outBufTDCTTT`
- `uint32_t nchannels`
- bool `gEnd`
- `std::string pair_lead_res` [8]
- `std::string pair_width_res` [16]
- `std::string trailead_edge_res` [4]

#### 7.13.1 Detailed Description

##### Author

Laurent Forthomme `laurent.forthomme@cern.ch`  
Bob Velghe `bob.velghe@cern.ch`

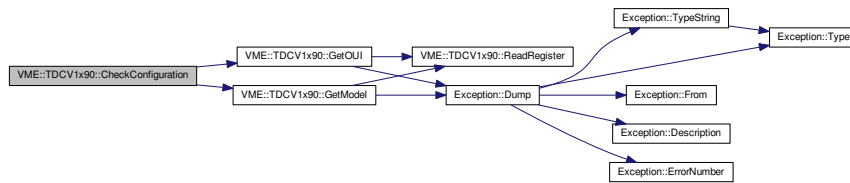
Jun 2010

```
7.13.2.1 VME::TDCV1x90::TDCV1x90 ( int32_t bhandle, uint32_t baseaddr, acq_mode acqm = TRIG_MATCH, det_mode
detm = TRAILLEAD )
```

### 7.13.3.1 void VME::TDCV1x90::abort ( )

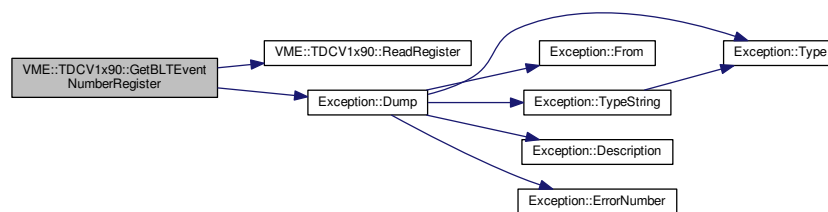
### 7.13.3.2 void VME::TDCV1x90::CheckConfiguration ( )

Here is the call graph for this function:



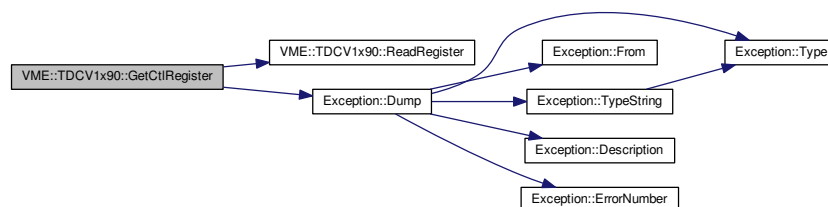
### 7.13.3.3 uint16\_t VME::TDCV1x90::GetBLTEventNumberRegister ( )

Here is the call graph for this function:



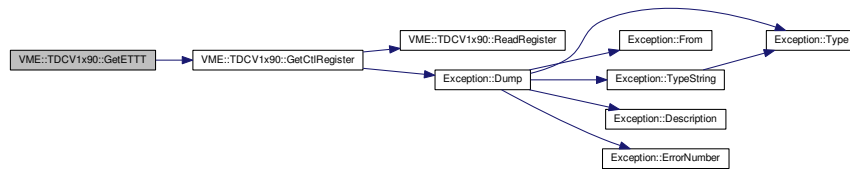
### 7.13.3.4 bool VME::TDCV1x90::GetCtlRegister ( ctl\_reg bit )

Here is the call graph for this function:



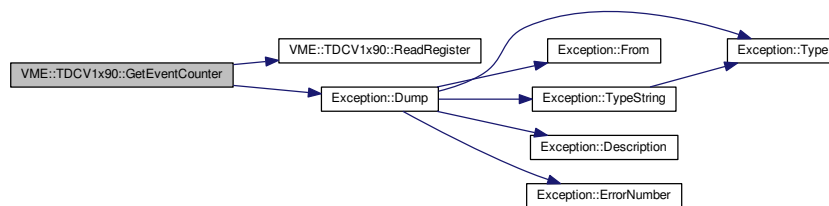
## 7.13.3.5 bool VME::TDCV1x90::GetETTT ( )

Here is the call graph for this function:



## 7.13.3.6 uint32\_t VME::TDCV1x90::GetEventCounter ( )

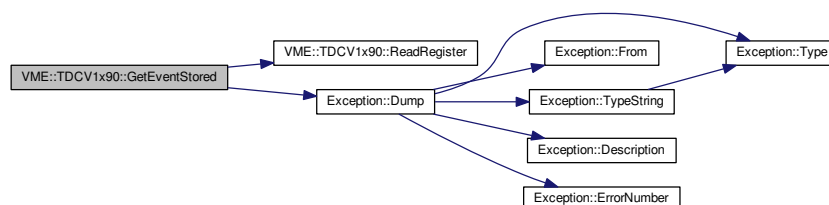
Here is the call graph for this function:



## 7.13.3.7 TDCEventCollection VME::TDCV1x90::GetEvents ( )

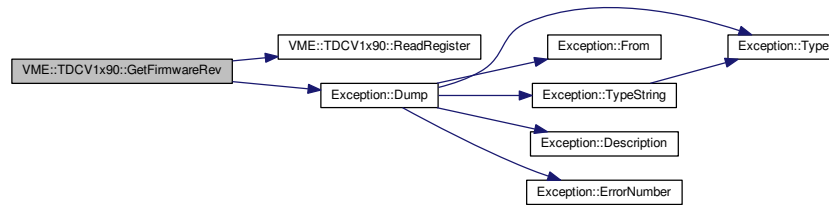
## 7.13.3.8 uint16\_t VME::TDCV1x90::GetEventStored ( )

Here is the call graph for this function:



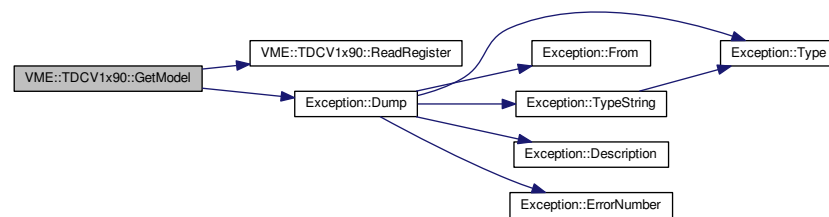
### 7.13.3.9 void VME::TDCV1x90::GetFirmwareRev ( )

Here is the call graph for this function:



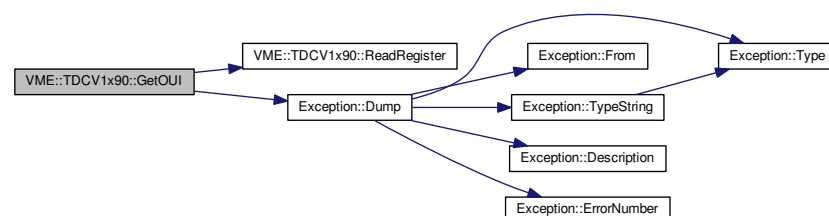
### 7.13.3.10 uint32\_t VME::TDCV1x90::GetModel ( )

Here is the call graph for this function:



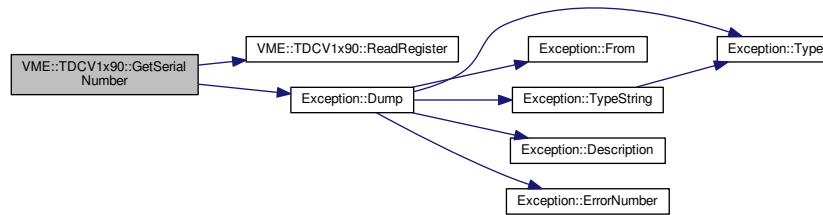
### 7.13.3.11 uint32\_t VME::TDCV1x90::GetOUI ( )

Here is the call graph for this function:



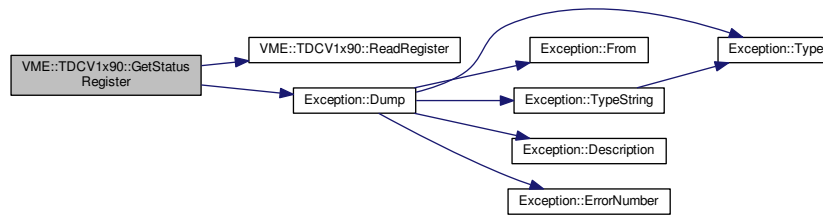
## 7.13.3.12 uint32\_t VME::TDCV1x90::GetSerialNumber ( )

Here is the call graph for this function:



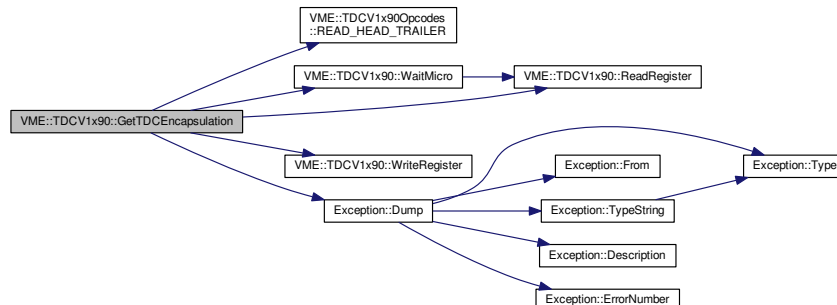
## 7.13.3.13 bool VME::TDCV1x90::GetStatusRegister ( stat\_reg bit )

Here is the call graph for this function:



## 7.13.3.14 bool VME::TDCV1x90::GetTDCEncapsulation ( )

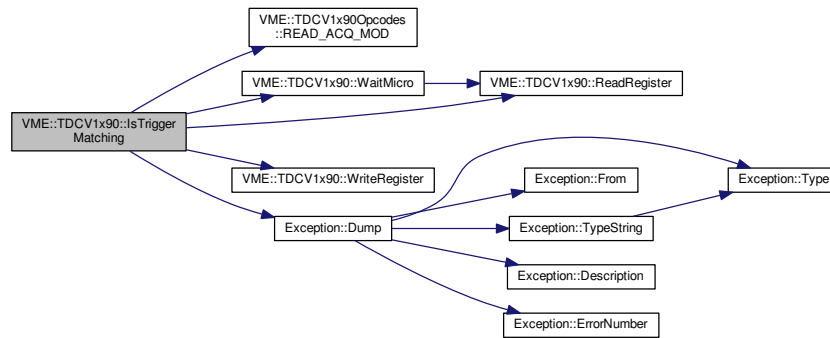
Here is the call graph for this function:



## 7.13.3.15 bool VME::TDCV1x90::HardwareReset ( )

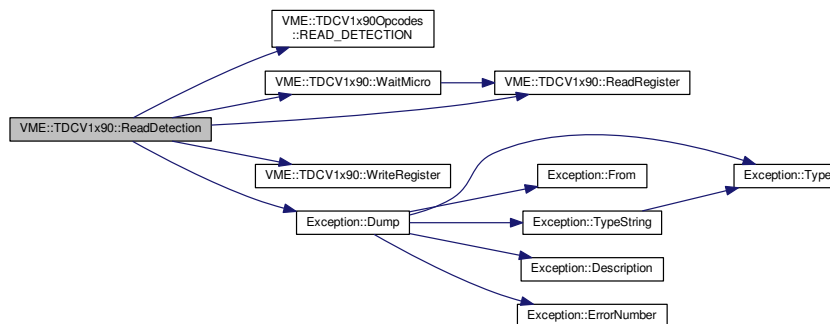
### 7.13.3.16 bool VME::TDCV1x90::IsTriggerMatching ( )

Here is the call graph for this function:



### 7.13.3.17 det\_mode VME::TDCV1x90::ReadDetection ( )

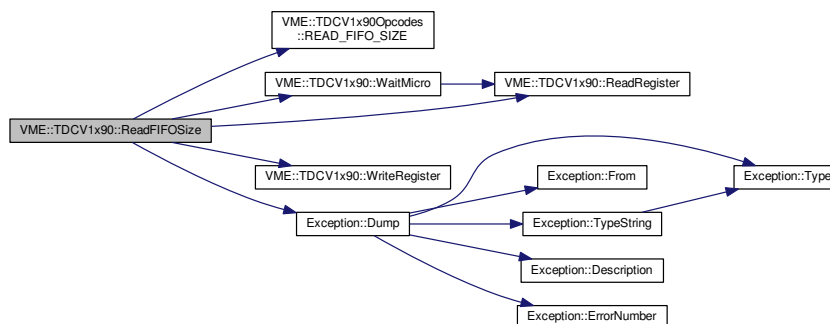
Here is the call graph for this function:





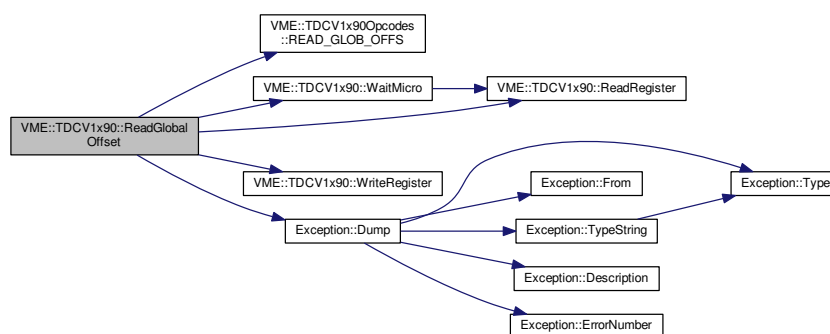
## 7.13.3.18 void VME::TDCV1x90::ReadFIFOSize ( )

Here is the call graph for this function:



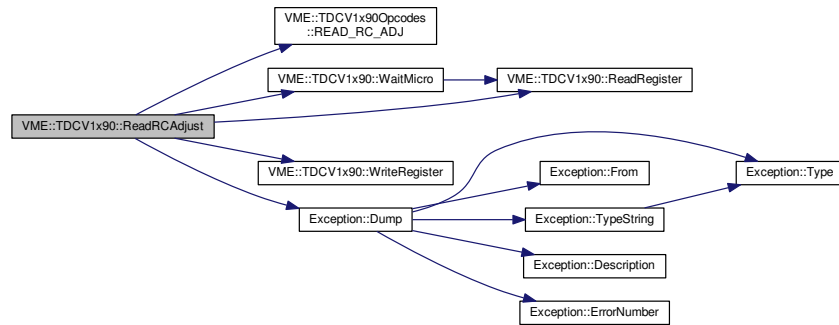
## 7.13.3.19 glob\_offs VME::TDCV1x90::ReadGlobalOffset ( )

Here is the call graph for this function:



### 7.13.3.20 uint16\_t VME::TDCV1x90::ReadRCAdjust ( int *tdc* )

Here is the call graph for this function:



### 7.13.3.21 void VME::TDCV1x90::ReadRegister ( mod\_reg *addr*, uint16\_t \* *data* )

Read on register.

Read a 16-bit word in the register

#### Parameters

in	<i>addr</i>	register
out	<i>data</i>	word

### 7.13.3.22 void VME::TDCV1x90::ReadRegister ( mod\_reg *addr*, uint32\_t \* *data* )

Read on register.

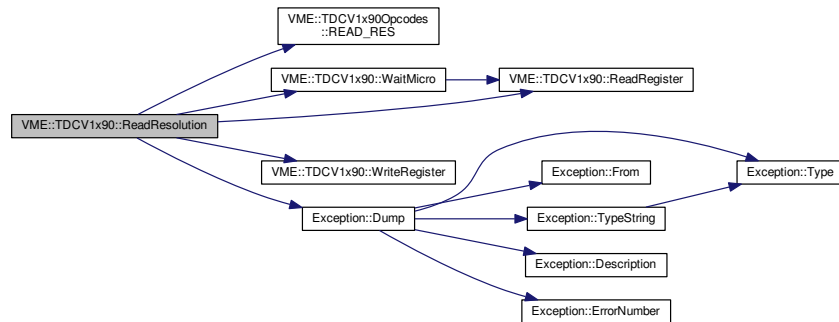
Read a 32-bit word in the register

#### Parameters

in	<i>addr</i>	register
out	<i>data</i>	word

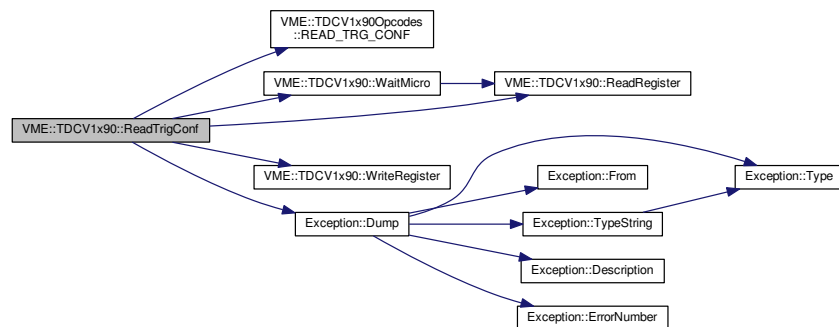
## 7.13.3.23 void VME::TDCV1x90::ReadResolution ( det\_mode det )

Here is the call graph for this function:



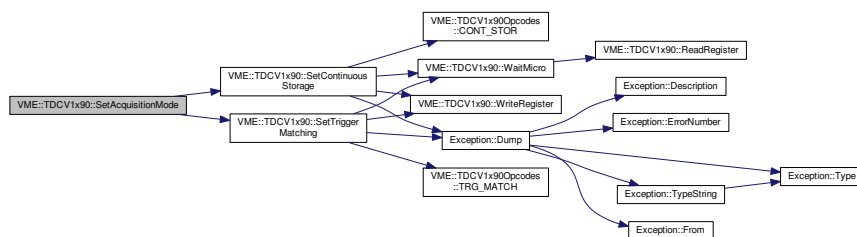
## 7.13.3.24 uint16\_t VME::TDCV1x90::ReadTrigConf ( trig\_conf type )

Here is the call graph for this function:



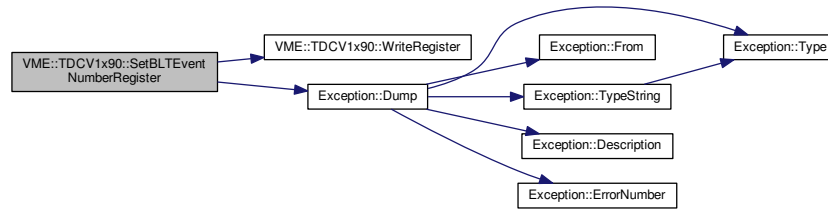
## 7.13.3.25 void VME::TDCV1x90::SetAcquisitionMode ( acq\_mode mode )

Here is the call graph for this function:



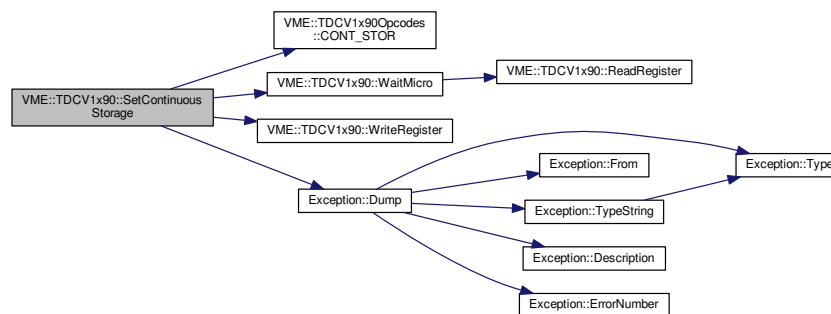
### 7.13.3.26 void VME::TDCV1x90::SetBLTEventNumberRegister ( uint16\_t value )

Here is the call graph for this function:



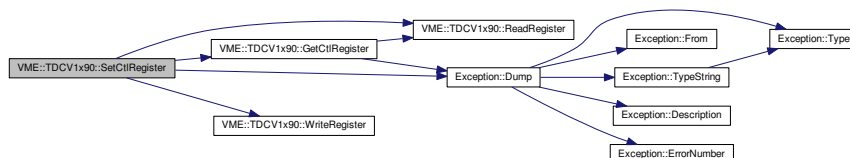
### 7.13.3.27 bool VME::TDCV1x90::SetContinuousStorage ( )

Here is the call graph for this function:



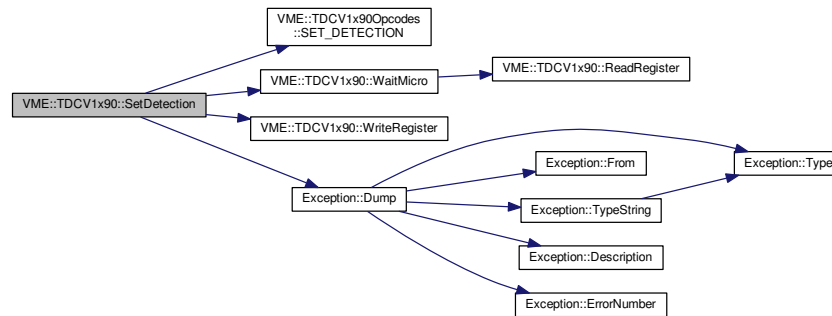
### 7.13.3.28 void VME::TDCV1x90::SetCtlRegister ( ctl\_reg reg, bool value )

Here is the call graph for this function:



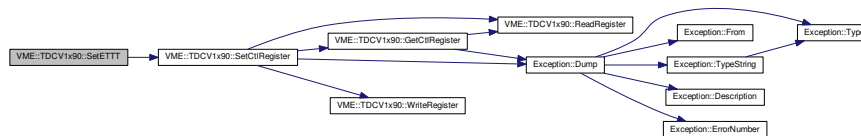
## 7.13.3.29 void VME::TDCV1x90::SetDetection ( det\_mode mode )

Here is the call graph for this function:



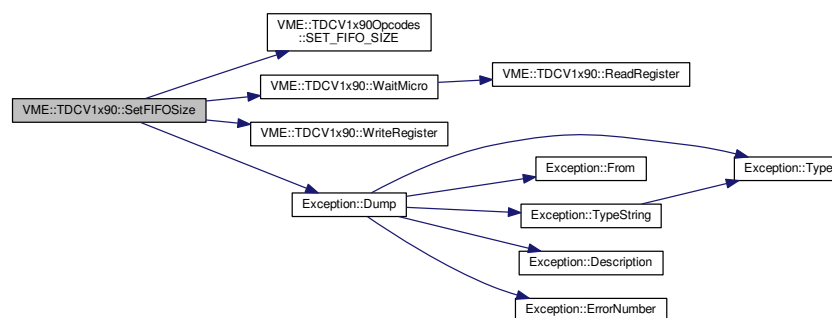
## 7.13.3.30 void VME::TDCV1x90::SetETTT ( bool mode )

Here is the call graph for this function:



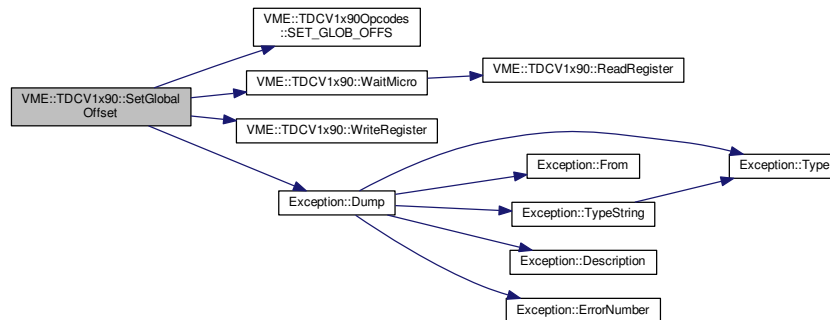
## 7.13.3.31 void VME::TDCV1x90::SetFIFOSize ( uint16\_t size )

Here is the call graph for this function:



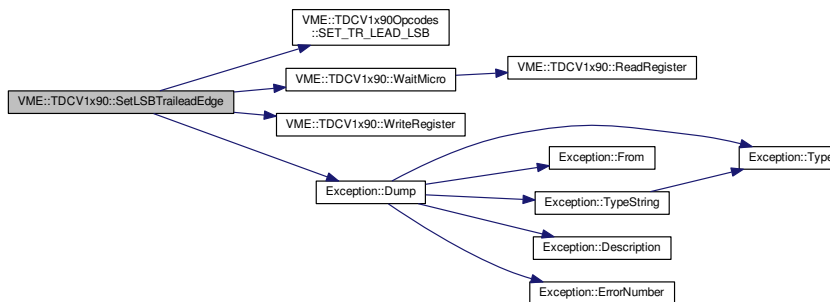
### 7.13.3.32 void VME::TDCV1x90::SetGlobalOffset ( uint16\_t word1, uint16\_t word2 )

Here is the call graph for this function:



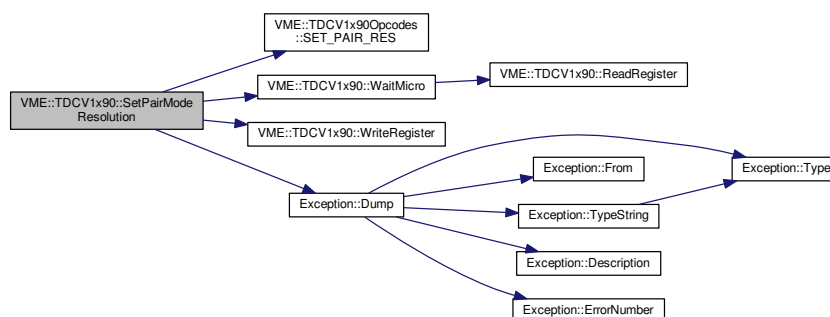
### 7.13.3.33 void VME::TDCV1x90::SetLSBTrailEdge ( traillead\_edge\_lsb conf )

Here is the call graph for this function:



### 7.13.3.34 void VME::TDCV1x90::SetPairModeResolution ( int lead\_time\_res, int pulse\_width\_res )

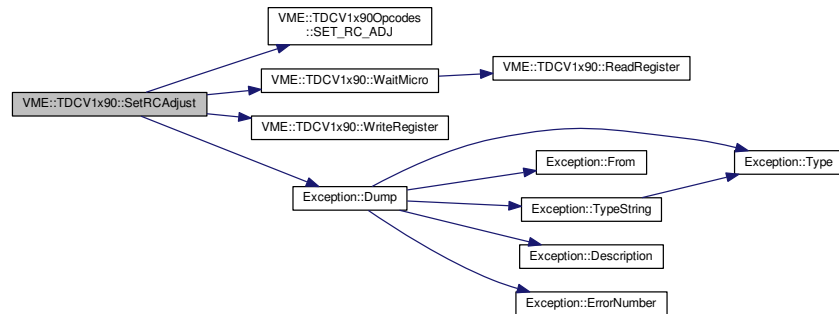
Here is the call graph for this function:



7.13.3.35 void VME::TDCV1x90::SetPol ( uint16\_t word )

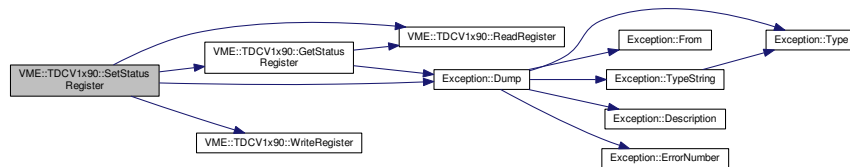
7.13.3.36 void VME::TDCV1x90::SetRCAdjust ( int tdc, uint16\_t value )

Here is the call graph for this function:



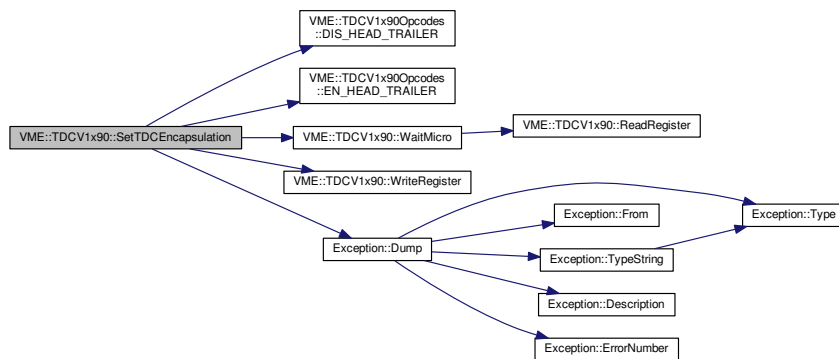
7.13.3.37 void VME::TDCV1x90::SetStatusRegister ( stat\_reg reg, bool value )

Here is the call graph for this function:



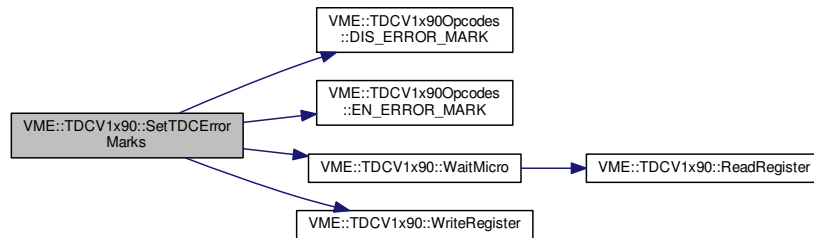
7.13.3.38 void VME::TDCV1x90::SetTDCEncapsulation ( bool mode )

Here is the call graph for this function:



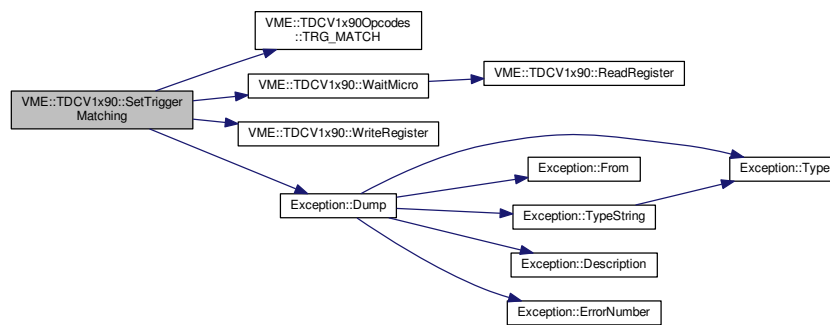
### 7.13.3.39 void VME::TDCV1x90::SetTDCErrorMarks ( bool mode )

Here is the call graph for this function:



### 7.13.3.40 bool VME::TDCV1x90::SetTriggerMatching ( )

Here is the call graph for this function:

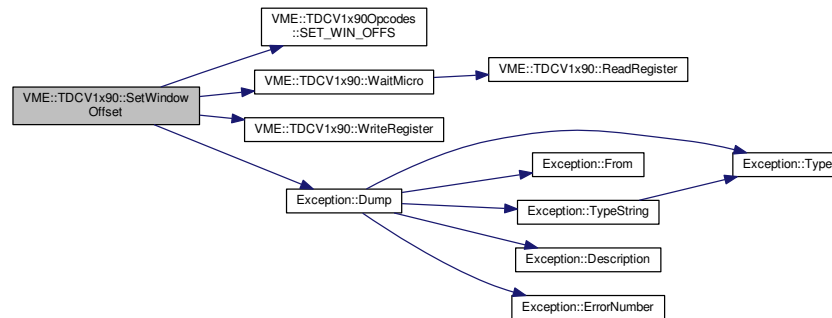


### 7.13.3.41 void VME::TDCV1x90::SetVerboseLevel ( unsigned short verb = 0 ) [inline]



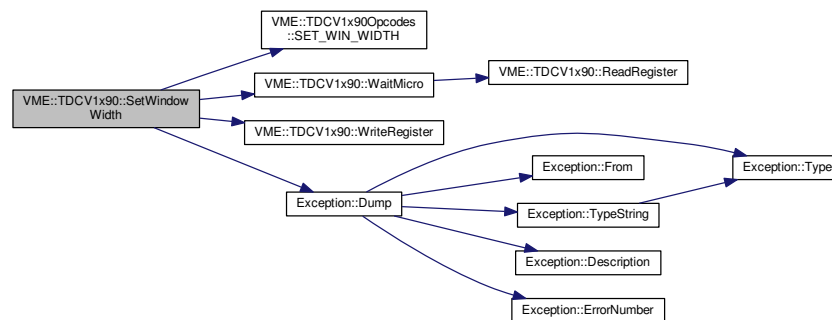
## 7.13.3.42 void VME::TDCV1x90::SetWindowOffset ( int16\_t offs )

Here is the call graph for this function:



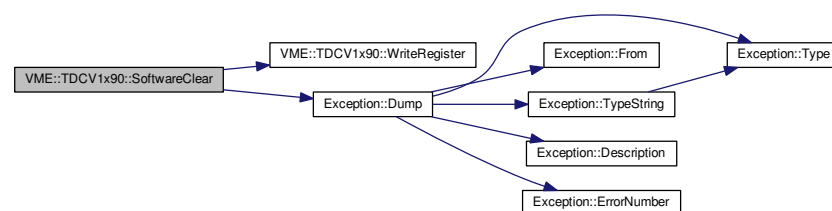
## 7.13.3.43 void VME::TDCV1x90::SetWindowWidth ( uint16\_t width )

Here is the call graph for this function:



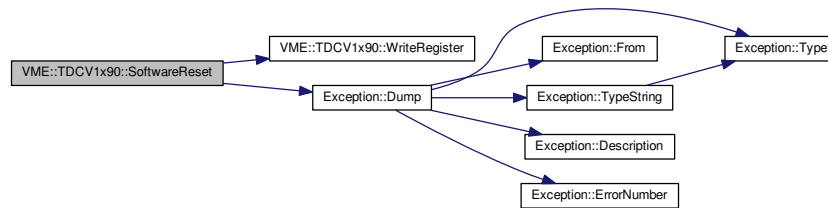
## 7.13.3.44 bool VME::TDCV1x90::SoftwareClear ( )

Here is the call graph for this function:



#### 7.13.3.45 `bool VME::TDCV1x90::SoftwareReset ( )`

Here is the call graph for this function:



#### 7.13.3.46 `bool VME::TDCV1x90::WaitMicro ( micro_handshake mode )`

Here is the call graph for this function:



#### 7.13.3.47 `void VME::TDCV1x90::WriteRegister ( mod_reg addr, uint16_t * data )`

Write on register.

Write a 16-bit word in the register

##### Parameters

in	<i>addr</i>	register
in	<i>data</i>	word

#### 7.13.3.48 `void VME::TDCV1x90::WriteRegister ( mod_reg addr, uint32_t * data )`

Write on register.

Write a 32-bit word in the register

##### Parameters

in	<i>addr</i>	register
in	<i>data</i>	word

### 7.13.4 Field Documentation

#### 7.13.4.1 `acq_mode VME::TDCV1x90::acqm [private]`

- 7.13.4.2 CVAddressModifier VME::TDCV1x90::am [private]
- 7.13.4.3 CVAddressModifier VME::TDCV1x90::am\_blt [private]
- 7.13.4.4 det\_mode VME::TDCV1x90::detm [private]
- 7.13.4.5 uint32\_t VME::TDCV1x90::fBaseAddr [private]
- 7.13.4.6 uint32\_t\* VME::TDCV1x90::fBuffer [private]
- 7.13.4.7 det\_mode VME::TDCV1x90::fDetMode [private]
- 7.13.4.8 int32\_t VME::TDCV1x90::fHandle [private]
- 7.13.4.9 unsigned short VME::TDCV1x90::fVerb [private]
- 7.13.4.10 bool VME::TDCV1x90::gEnd [private]
- 7.13.4.11 uint32\_t VME::TDCV1x90::nchannels [private]
- 7.13.4.12 bool VME::TDCV1x90::outBufTDCErr [private]
- 7.13.4.13 bool VME::TDCV1x90::outBufTDCHeadTrail [private]
- 7.13.4.14 bool VME::TDCV1x90::outBufTDCTTT [private]
- 7.13.4.15 std::string VME::TDCV1x90::pair\_lead\_res[8] [private]
- 7.13.4.16 std::string VME::TDCV1x90::pair\_width\_res[16] [private]
- 7.13.4.17 std::string VME::TDCV1x90::trailead\_edge\_res[4] [private]

The documentation for this class was generated from the following files:

- include/VME\_TDCV1x90.h
- src/VME\_TDCV1x90.cpp

## 7.14 VME::trailead\_t Struct Reference

```
#include <VME_TDCV1x90.h>
```

### Data Fields

- uint32\_t [event\\_count](#)
- int [total\\_hits](#) [16]
- std::multimap< int32\_t, int32\_t > [leading](#)
- std::multimap< int32\_t, int32\_t > [trailing](#)
- uint32\_t [ett](#)

### 7.14.1 Field Documentation

- 7.14.1.1 uint32\_t VME::trailead\_t::ett

7.14.1.2 `uint32_t VME::trailead_t::event_count`

7.14.1.3 `std::multimap<int32_t,int32_t> VME::trailead_t::leading`

7.14.1.4 `int VME::trailead_t::total_hits[16]`

7.14.1.5 `std::multimap<int32_t,int32_t> VME::trailead_t::trailing`

The documentation for this struct was generated from the following file:

- `include/VME_TDCV1x90.h`

# Index

- ~BridgeVx718
  - VME::BridgeVx718, [20](#)
- ~Client
  - Client, [22](#)
- ~Exception
  - Exception, [26](#)
- ~FileReader
  - FileReader, [29](#)
- ~Message
  - Message, [34](#)
- ~Messenger
  - Messenger, [37](#)
- ~Socket
  - Socket, [44](#)
- ~SocketMessage
  - SocketMessage, [52](#)
- ~TDCEvent
  - VME::TDCEvent, [56](#)
- ~TDCV1x90
  - VME::TDCV1x90, [63](#)
- ALIGN64
  - VME, [12](#)
- ALM\_FULLL
  - VME, [14](#)
- AUTOLOAD\_DEF\_CONFI
  - VME::TDCV1x90Opcodes, [16](#)
- AUTOLOAD\_USER\_CONF
  - VME::TDCV1x90Opcodes, [16](#)
- abort
  - VME::TDCV1x90, [63](#)
- AcceptConnections
  - Socket, [44](#)
- acq\_mode
  - VME, [12](#)
- acqm
  - VME::TDCV1x90, [78](#)
- AddClient
  - Messenger, [37](#)
- am
  - VME::TDCV1x90, [78](#)
- am\_blt
  - VME::TDCV1x90, [79](#)
- Announce
  - Client, [23](#)
- BERR\_FLAG
  - VME, [14](#)
- BERREN
  - VME, [12](#)
- BLTEventNumber
  - VME, [13](#)
- Bind
  - Socket, [44](#)
- BridgeVx718
  - VME::BridgeVx718, [20](#)
- Broadcast
  - Messenger, [37](#)
- CLEAR\_KEEP\_TOKEN
  - VME::TDCV1x90Opcodes, [16](#)
- CLIENT
  - Socket communication objects, [9](#)
- COMPENSATION\_ENABLE
  - VME, [12](#)
- CONT\_STOR
  - VME::TDCV1x90Opcodes, [16](#)
- CONT\_STORAGE
  - VME, [12](#)
- CheckConfiguration
  - VME::TDCV1x90, [63](#)
- Client, [21](#)
  - ~Client, [22](#)
  - Announce, [23](#)
  - Client, [22](#)
  - Connect, [23](#)
  - Disconnect, [24](#)
  - fClientId, [25](#)
  - flsConnected, [25](#)
  - GetType, [24](#)
  - ParseMessage, [24](#)
  - Receive, [24](#)
  - Send, [25](#)
- coarse
  - VME::glob\_offs, [30](#)
- Configure
  - Socket, [44](#)
- Connect
  - Client, [23](#)
  - Messenger, [38](#)
- Control
  - VME, [13](#)
- Create
  - Socket, [45](#)
- ctl\_reg
  - VME, [12](#)
- DATA\_READY
  - VME, [14](#)
- DETECTOR

- Socket communication objects, 9
- DIS\_ALL\_CHANNEL
  - VME::TDCV1x90Opcodes, 16
- DIS\_CHANNEL
  - VME::TDCV1x90Opcodes, 16
- DIS\_ERROR\_BYPASS
  - VME::TDCV1x90Opcodes, 16
- DIS\_ERROR\_MARK
  - VME::TDCV1x90Opcodes, 16
- DIS\_HEAD\_TRAILER
  - VME::TDCV1x90Opcodes, 16
- DIS\_SUB\_TRG
  - VME::TDCV1x90Opcodes, 16
- Decode
  - HTTPMessage, 32
- Description
  - Exception, 27
- det\_mode
  - VME, 12
- detm
  - VME::TDCV1x90, 79
- Disconnect
  - Client, 24
  - Messenger, 38
- DisconnectClient
  - Messenger, 39
- Dump
  - Exception, 27
  - HTTPMessage, 32
  - Message, 34
  - SocketMessage, 52
- DumpConnected
  - Socket, 45
- EMPTY\_EVENT
  - VME, 12
- EN\_ALL\_CHANNEL
  - VME::TDCV1x90Opcodes, 16
- EN\_CHANNEL
  - VME::TDCV1x90Opcodes, 16
- EN\_ERROR\_BYPASS
  - VME::TDCV1x90Opcodes, 16
- EN\_ERROR\_MARK
  - VME::TDCV1x90Opcodes, 16
- EN\_HEAD\_TRAILER
  - VME::TDCV1x90Opcodes, 16
- EN\_SUB\_TRG
  - VME::TDCV1x90Opcodes, 16
- ERROR0
  - VME, 14
- ERROR1
  - VME, 14
- ERROR2
  - VME, 14
- ERROR3
  - VME, 14
- ETTT
  - VME::TDCEvent, 56
- EVENT\_FIFO\_ENABLE
  - VME, 12
- EXTENDED\_TRIGGER\_TIME\_TAG\_ENABLE
  - VME, 12
- EXTRA\_SEARCH\_WIN\_WIDTH
  - VME, 14
- Encode
  - HTTPMessage, 32
- ErrorNumber
  - Exception, 27
- ettt
  - VME::trailead\_t, 79
- event\_count
  - VME::trailead\_t, 79
- EventCounter
  - VME, 13
- EventFIFO
  - VME, 13
- EventFIFOStatusRegister
  - VME, 13
- EventFIFOStoredRegister
  - VME, 13
- EventStored
  - VME, 13
- EventType
  - VME::TDCEvent, 56
- Exception, 25
  - ~Exception, 26
  - Description, 27
  - Dump, 27
  - ErrorNumber, 27
  - Exception, 26
  - fDescription, 27
  - fErrorNumber, 27
  - fFrom, 27
  - fType, 27
  - From, 27
  - Type, 27
  - TypeString, 27
- fAddress
  - Socket, 47
- fBaseAddr
  - VME::TDCV1x90, 79
- fBuffer
  - Socket, 47
  - VME::TDCV1x90, 79
- fClientId
  - Client, 25
- fDescription
  - Exception, 27
- fDetMode
  - VME::TDCV1x90, 79
- fErrorNumber
  - Exception, 27
- fFile
  - FileReader, 29
- fFrom
  - Exception, 27
- fHandle

- VME::BridgeVx718, 20
- VME::TDCV1x90, 79
- fHeader
  - FileReader, 29
- flsConnected
  - Client, 25
- fMaster
  - Socket, 47
- fMessage
  - SocketMessage, 54
- fNumAttempts
  - Messenger, 41
- fOriginalString
  - HTTPMessage, 32
- fPort
  - Socket, 47
- fPortMapping
  - VME::BridgeVx718, 21
- fReadFds
  - Socket, 47
- fSocketId
  - Socket, 47
- fSocketsConnected
  - Socket, 47
- fString
  - Message, 34
- fType
  - Exception, 27
- FULL
  - VME, 14
- fVerb
  - VME::TDCV1x90, 79
- fWS
  - HTTPMessage, 32
  - Messenger, 41
- fWord
  - VME::TDCEvent, 60
- FetchMessage
  - Socket, 45
- file\_header\_t, 28
  - magic, 28
  - num\_hptdc, 28
  - run\_id, 28
  - spill\_id, 28
- FileReader, 28
  - ~FileReader, 29
  - fFile, 29
  - fHeader, 29
  - FileReader, 29
  - GetNextEvent, 29
  - GetNumTDCs, 29
- Filler
  - VME::TDCEvent, 56
- fine
  - VME::glob\_offs, 30
- FirmwareRev
  - VME, 13
- From
  - Exception, 27
- gEnd
  - VME::TDCV1x90, 79
- GeoAddress
  - VME, 13
- GetBLTEventNumberRegister
  - VME::TDCV1x90, 64
- GetBunchId
  - VME::TDCEvent, 56
- GetChannelId
  - VME::TDCEvent, 56
- GetCtlRegister
  - VME::TDCV1x90, 64
- GetETTT
  - VME::TDCEvent, 57
  - VME::TDCV1x90, 64
- GetErrorFlags
  - VME::TDCEvent, 57
- GetEventCount
  - VME::TDCEvent, 57
- GetEventCounter
  - VME::TDCV1x90, 65
- GetEventId
  - VME::TDCEvent, 58
- GetEventStored
  - VME::TDCV1x90, 65
- GetEvents
  - VME::TDCV1x90, 65
- GetFirmwareRev
  - VME::TDCV1x90, 65
- GetGeo
  - VME::TDCEvent, 58
- GetHandle
  - VME::BridgeVx718, 20
- GetIntValue
  - SocketMessage, 52
- GetKey
  - HTTPMessage, 32
  - Message, 34
  - SocketMessage, 52
- GetLeadingTime
  - VME::TDCEvent, 58
- GetModel
  - VME::TDCV1x90, 66
- GetNextEvent
  - FileReader, 29
- GetNumTDCs
  - FileReader, 29
- GetOUI
  - VME::TDCV1x90, 66
- GetPort
  - Socket, 45
- GetSerialNumber
  - VME::TDCV1x90, 66
- GetSocketId
  - Socket, 45
- GetSocketType
  - Socket, 45

- GetStatusRegister
  - VME::TDCV1x90, 67
- GetString
  - Message, 34
  - SocketMessage, 53
- GetTDCEncapsulation
  - VME::TDCV1x90, 67
- GetTDCId
  - VME::TDCEvent, 59
- GetTrailingTime
  - VME::TDCEvent, 59
- GetType
  - Client, 24
  - Messenger, 39
  - VME::TDCEvent, 59
- GetValue
  - SocketMessage, 53
- GetVectorValue
  - SocketMessage, 53
- GetWidth
  - VME::TDCEvent, 59
- GetWordCount
  - VME::TDCEvent, 60
- GlobalHeader
  - VME::TDCEvent, 56
- GlobalTrailer
  - VME::TDCEvent, 56
- HEADER\_EN
  - VME, 14
- HTTPMessage, 30
  - Decode, 32
  - Dump, 32
  - Encode, 32
  - fOriginalString, 32
  - fWS, 32
  - GetKey, 32
  - HTTPMessage, 31, 32
- HardwareReset
  - VME::TDCV1x90, 67
- INVALID
  - Socket communication objects, 9
- InputConf
  - VME::BridgeVx718, 20
- InputRead
  - VME::BridgeVx718, 20
- InterruptLevel
  - VME, 13
- InterruptVector
  - VME, 13
- IsFromWeb
  - Message, 34
- IsTrailing
  - VME::TDCEvent, 60
- IsTriggerMatching
  - VME::TDCV1x90, 67
- IsWebSocket
  - Socket, 45
- kSoftwareClear
  - VME, 13
- LOAD\_DEF\_CONFIG
  - VME::TDCV1x90Opcodes, 16
- LOAD\_USER\_CONFIG
  - VME::TDCV1x90Opcodes, 16
- leading
  - VME::trailead\_t, 80
- Listen
  - Socket, 45
- MASTER
  - Socket communication objects, 9
- MATCH\_WIN\_WIDTH
  - VME, 14
- MCSTBase
  - VME, 13
- MCSTControl
  - VME, 13
- magic
  - file\_header\_t, 28
- Message, 33
  - ~Message, 34
  - Dump, 34
  - fString, 34
  - GetKey, 34
  - GetString, 34
  - IsFromWeb, 34
  - Message, 34
- Messenger, 35
  - ~Messenger, 37
  - AddClient, 37
  - Broadcast, 37
  - Connect, 38
  - Disconnect, 38
  - DisconnectClient, 39
  - fNumAttempts, 41
  - fWS, 41
  - GetType, 39
  - Messenger, 36
  - ProcessMessage, 39
  - Receive, 40
  - Send, 40
  - SwitchClientType, 41
- Micro
  - VME, 13
- micro\_handshake
  - VME, 12
- MicroHandshake
  - VME, 13
- mod\_reg
  - VME, 13
- ModuleReset
  - VME, 13
- nchannels
  - VME::TDCV1x90, 79
- num\_hptdc



- file\_header\_t, 28
- OLEADING
  - VME, 12
- OTRAILING
  - VME, 12
- Object
  - SocketMessage, 53
- outBufTDCErr
  - VME::TDCV1x90, 79
- outBufTDCHeadTrail
  - VME::TDCV1x90, 79
- outBufTDC\_TTT
  - VME::TDCV1x90, 79
- OutputConf
  - VME::BridgeVx718, 20
- OutputOff
  - VME::BridgeVx718, 20
- OutputOn
  - VME::BridgeVx718, 20
- PAIR
  - VME, 12
- PAIRED
  - VME, 14
- PURG
  - VME, 14
- pair\_lead\_res
  - VME::TDCV1x90, 79
- pair\_width\_res
  - VME::TDCV1x90, 79
- ParseMessage
  - Client, 24
- PrepareConnection
  - Socket, 46
- ProcessMessage
  - Messenger, 39
- r100ps
  - VME, 14
- r200ps
  - VME, 14
- r25ps
  - VME, 14
- r800ps
  - VME, 14
- READ\_ACQ\_MOD
  - VME::TDCV1x90Opcodes, 16
- READ\_COMPENSATION\_SRAM\_ENABLE
  - VME, 12
- READ\_DEAD\_TIME
  - VME::TDCV1x90Opcodes, 16
- READ\_DETECTION
  - VME::TDCV1x90Opcodes, 16
- READ\_EN\_PATTERN
  - VME::TDCV1x90Opcodes, 16
- READ\_EN\_PATTERN32
  - VME::TDCV1x90Opcodes, 16
- READ\_ERROR\_TYPES
  - VME::TDCV1x90Opcodes, 16
- READ\_EVENT\_SIZE
  - VME::TDCV1x90Opcodes, 16
- READ\_FIFO\_SIZE
  - VME::TDCV1x90Opcodes, 16
- READ\_GLOB\_OFFS
  - VME::TDCV1x90Opcodes, 16
- READ\_HEAD\_TRAILER
  - VME::TDCV1x90Opcodes, 16
- READ\_OK
  - VME, 13
- READ\_RC\_ADJ
  - VME::TDCV1x90Opcodes, 17
- READ\_RES
  - VME::TDCV1x90Opcodes, 17
- READ\_TRG\_CONF
  - VME::TDCV1x90Opcodes, 17
- REJECT\_MARGIN
  - VME, 14
- RES\_1
  - VME, 14
- RES\_2
  - VME, 14
- ROMBoard0
  - VME, 13
- ROMBoard1
  - VME, 13
- ROMBoard2
  - VME, 13
- ROMOui0
  - VME, 13
- ROMOui1
  - VME, 13
- ROMOui2
  - VME, 13
- ROMRevis0
  - VME, 13
- ROMRevis1
  - VME, 13
- ROMRevis2
  - VME, 13
- ROMRevis3
  - VME, 13
- ROMSerNum0
  - VME, 13
- ROMSerNum1
  - VME, 13
- ReadDetection
  - VME::TDCV1x90, 68
- ReadFIFOSize
  - VME::TDCV1x90, 68
- ReadGlobalOffset
  - VME::TDCV1x90, 69
- ReadRCAdjust
  - VME::TDCV1x90, 69
- ReadRegister
  - VME::TDCV1x90, 70
- ReadResolution

- VME::TDCV1x90, [70](#)
- ReadTrigConf
  - VME::TDCV1x90, [71](#)
- Receive
  - Client, [24](#)
  - Messenger, [40](#)
- run\_id
  - file\_header\_t, [28](#)
- SAVE\_RC\_ADJ
  - VME::TDCV1x90Opcodes, [17](#)
- SAVE\_USER\_CONFIG
  - VME::TDCV1x90Opcodes, [17](#)
- SET\_DEAD\_TIME
  - VME::TDCV1x90Opcodes, [17](#)
- SET\_DETECTION
  - VME::TDCV1x90Opcodes, [17](#)
- SET\_ERROR\_TYPES
  - VME::TDCV1x90Opcodes, [17](#)
- SET\_EVENT\_SIZE
  - VME::TDCV1x90Opcodes, [17](#)
- SET\_FIFO\_SIZE
  - VME::TDCV1x90Opcodes, [17](#)
- SET\_GLOB\_OFFS
  - VME::TDCV1x90Opcodes, [17](#)
- SET\_KEEP\_TOKEN
  - VME::TDCV1x90Opcodes, [17](#)
- SET\_PAIR\_RES
  - VME::TDCV1x90Opcodes, [17](#)
- SET\_RC\_ADJ
  - VME::TDCV1x90Opcodes, [17](#)
- SET\_REJ\_MARGIN
  - VME::TDCV1x90Opcodes, [17](#)
- SET\_SW\_MARGIN
  - VME::TDCV1x90Opcodes, [17](#)
- SET\_TR\_LEAD\_LSB
  - VME::TDCV1x90Opcodes, [17](#)
- SET\_WIN\_OFFS
  - VME::TDCV1x90Opcodes, [17](#)
- SET\_WIN\_WIDTH
  - VME::TDCV1x90Opcodes, [17](#)
- SelectConnections
  - Socket, [46](#)
- Send
  - Client, [25](#)
  - Messenger, [40](#)
- SendMessage
  - Socket, [46](#)
- SetAcquisitionMode
  - VME::TDCV1x90, [71](#)
- SetBLTEventNumberRegister
  - VME::TDCV1x90, [71](#)
- SetContinuousStorage
  - VME::TDCV1x90, [72](#)
- SetCtlRegister
  - VME::TDCV1x90, [72](#)
- SetDetection
  - VME::TDCV1x90, [72](#)
- SetETTT
  - VME::TDCV1x90, [73](#)
- SetFIFOSize
  - VME::TDCV1x90, [73](#)
- SetGlobalOffset
  - VME::TDCV1x90, [73](#)
- SetKeyValue
  - SocketMessage, [53](#), [54](#)
- SetLSBTrailEdge
  - VME::TDCV1x90, [74](#)
- SetPairModeResolution
  - VME::TDCV1x90, [74](#)
- SetPol
  - VME::TDCV1x90, [75](#)
- SetPort
  - Socket, [46](#)
- SetRCAdjust
  - VME::TDCV1x90, [75](#)
- SetSocketId
  - Socket, [46](#)
- SetStatusRegister
  - VME::TDCV1x90, [75](#)
- SetTDCEncapsulation
  - VME::TDCV1x90, [75](#)
- SetTDCErrorMarks
  - VME::TDCV1x90, [75](#)
- SetTriggerMatching
  - VME::TDCV1x90, [76](#)
- SetVerboseLevel
  - VME::TDCV1x90, [76](#)
- SetWindowOffset
  - VME::TDCV1x90, [76](#)
- SetWindowWidth
  - VME::TDCV1x90, [77](#)
- SetWord
  - VME::TDCEvent, [60](#)
- Socket, [42](#)
  - ~Socket, [44](#)
  - AcceptConnections, [44](#)
  - Bind, [44](#)
  - Configure, [44](#)
  - Create, [45](#)
  - DumpConnected, [45](#)
  - fAddress, [47](#)
  - fBuffer, [47](#)
  - fMaster, [47](#)
  - fPort, [47](#)
  - fReadFds, [47](#)
  - fSocketId, [47](#)
  - fSocketsConnected, [47](#)
  - FetchMessage, [45](#)
  - GetPort, [45](#)
  - GetSocketId, [45](#)
  - GetSocketType, [45](#)
  - IsWebSocket, [45](#)
  - Listen, [45](#)
  - PrepareConnection, [46](#)
  - SelectConnections, [46](#)
  - SendMessage, [46](#)

- SetPort, [46](#)
- SetSocketId, [46](#)
- Socket, [44](#)
- SocketCollection, [44](#)
- Start, [46](#)
- Stop, [47](#)
- Socket communication objects, [9](#)
  - CLIENT, [9](#)
  - DETECTOR, [9](#)
  - INVALID, [9](#)
  - MASTER, [9](#)
  - SocketType, [9](#)
  - WEBSOCKET\_CLIENT, [9](#)
- SocketCollection
  - Socket, [44](#)
- SocketMessage, [48](#)
  - ~SocketMessage, [52](#)
  - Dump, [52](#)
  - fMessage, [54](#)
  - GetIntValue, [52](#)
  - GetKey, [52](#)
  - GetString, [53](#)
  - GetValue, [53](#)
  - GetVectorValue, [53](#)
  - Object, [53](#)
  - SetKeyValue, [53](#), [54](#)
  - SocketMessage, [50–52](#)
  - String, [54](#)
- SocketType
  - Socket communication objects, [9](#)
- SoftwareClear
  - VME::TDCV1x90, [77](#)
- SoftwareReset
  - VME::TDCV1x90, [77](#)
- spill\_id
  - file\_header\_t, [28](#)
- Start
  - Socket, [46](#)
- stat\_reg
  - VME, [13](#)
- Status
  - VME, [13](#)
- Stop
  - Socket, [47](#)
- String
  - SocketMessage, [54](#)
- SwitchClientType
  - Messenger, [41](#)
- TDCErrors
  - VME::TDCEvent, [56](#)
- TDCEvent
  - VME::TDCEvent, [56](#)
- TDCEventCollection
  - VME, [12](#)
- TDCHdr
  - VME::TDCEvent, [56](#)
- TDCMeasurement
  - VME::TDCEvent, [56](#)
- TDCTrailer
  - VME::TDCEvent, [56](#)
- TDCV1x90
  - VME::TDCV1x90, [63](#)
- TERM
  - VME, [12](#)
- TERM\_ON
  - VME, [14](#)
- TERM\_SW
  - VME, [12](#)
- TEST\_FIFO\_ENABLE
  - VME, [12](#)
- TRAILHEAD
  - VME, [12](#)
- TRG\_MATCH
  - VME, [14](#)
  - VME::TDCV1x90Opcodes, [17](#)
- TRIG\_MATCH
  - VME, [12](#)
- TRIG\_TIME\_SUB
  - VME, [14](#)
- TRIGGER\_LOST
  - VME, [14](#)
- total\_hits
  - VME::trailead\_t, [80](#)
- trailead\_edge\_lsb
  - VME, [14](#)
- trailead\_edge\_res
  - VME::TDCV1x90, [79](#)
- trailing
  - VME::trailead\_t, [80](#)
- trig\_conf
  - VME, [14](#)
- Type
  - Exception, [27](#)
- TypeString
  - Exception, [27](#)
- VME, [11](#)
  - ALIGN64, [12](#)
  - ALM\_FULL, [14](#)
  - acq\_mode, [12](#)
  - BERR\_FLAG, [14](#)
  - BERREN, [12](#)
  - BLTEventNumber, [13](#)
  - COMPENSATION\_ENABLE, [12](#)
  - CONT\_STORAGE, [12](#)
  - Control, [13](#)
  - ctl\_reg, [12](#)
  - DATA\_READY, [14](#)
  - det\_mode, [12](#)
  - EMPTY\_EVENT, [12](#)
  - ERROR0, [14](#)
  - ERROR1, [14](#)
  - ERROR2, [14](#)
  - ERROR3, [14](#)
  - EVENT\_FIFO\_ENABLE, [12](#)
  - EXTENDED\_TRIGGER\_TIME\_TAG\_ENABLE, [12](#)
  - EXTRA\_SEARCH\_WIN\_WIDTH, [14](#)

EventCounter, 13  
 EventFIFO, 13  
 EventFIFOStatusRegister, 13  
 EventFIFOStoredRegister, 13  
 EventStored, 13  
 FULL, 14  
 FirmwareRev, 13  
 GeoAddress, 13  
 HEADER\_EN, 14  
 InterruptLevel, 13  
 InterruptVector, 13  
 kSoftwareClear, 13  
 MATCH\_WIN\_WIDTH, 14  
 MCSTBase, 13  
 MCSTControl, 13  
 Micro, 13  
 micro\_handshake, 12  
 MicroHandshake, 13  
 mod\_reg, 13  
 ModuleReset, 13  
 OLEADING, 12  
 OTRAILING, 12  
 PAIR, 12  
 PAIRED, 14  
 PURG, 14  
 r100ps, 14  
 r200ps, 14  
 r25ps, 14  
 r800ps, 14  
 READ\_COMPENSATION\_SRAM\_ENABLE, 12  
 READ\_OK, 13  
 REJECT\_MARGIN, 14  
 RES\_1, 14  
 RES\_2, 14  
 ROMBoard0, 13  
 ROMBoard1, 13  
 ROMBoard2, 13  
 ROMOui0, 13  
 ROMOui1, 13  
 ROMOui2, 13  
 ROMRevis0, 13  
 ROMRevis1, 13  
 ROMRevis2, 13  
 ROMRevis3, 13  
 ROMSerNum0, 13  
 ROMSerNum1, 13  
 stat\_reg, 13  
 Status, 13  
 TDCEventCollection, 12  
 TERM, 12  
 TERM\_ON, 14  
 TERM\_SW, 12  
 TEST\_FIFO\_ENABLE, 12  
 TRAILHEAD, 12  
 TRG\_MATCH, 14  
 TRIG\_MATCH, 12  
 TRIG\_TIME\_SUB, 14  
 TRIGGER\_LOST, 14  
 trailead\_edge\_lsb, 14  
 trig\_conf, 14  
 WIN\_OFFSET, 14  
 WRITE\_OK, 13  
 VME::BridgeVx718, 19  
   ~BridgeVx718, 20  
   BridgeVx718, 20  
   fHandle, 20  
   fPortMapping, 21  
   GetHandle, 20  
   InputConf, 20  
   InputRead, 20  
   OutputConf, 20  
   OutputOff, 20  
   OutputOn, 20  
 VME::TDCEvent, 54  
   ~TDCEvent, 56  
   ETTT, 56  
   EventType, 56  
   fWord, 60  
   Filler, 56  
   GetBunchId, 56  
   GetChannelId, 56  
   GetETTT, 57  
   GetErrorFlags, 57  
   GetEventCount, 57  
   GetEventId, 58  
   GetGeo, 58  
   GetLeadingTime, 58  
   GetTDCId, 59  
   GetTrailingTime, 59  
   GetType, 59  
   GetWidth, 59  
   GetWordCount, 60  
   GlobalHeader, 56  
   GlobalTrailer, 56  
   IsTrailing, 60  
   SetWord, 60  
   TDCErrors, 56  
   TDCEvent, 56  
   TDCHeader, 56  
   TDCMeasurement, 56  
   TDCTrailer, 56  
 VME::TDCV1x90, 61  
   ~TDCV1x90, 63  
   abort, 63  
   acqm, 78  
   am, 78  
   am\_blt, 79  
   CheckConfiguration, 63  
   detm, 79  
   fBaseAddr, 79  
   fBuffer, 79  
   fDetMode, 79  
   fHandle, 79  
   fVerb, 79  
   gEnd, 79  
   GetBLTEventNumberRegister, 64

- GetCtlRegister, 64
- GetETTT, 64
- GetEventCounter, 65
- GetEventStored, 65
- GetEvents, 65
- GetFirmwareRev, 65
- GetModel, 66
- GetOUI, 66
- GetSerialNumber, 66
- GetStatusRegister, 67
- GetTDCEncapsulation, 67
- HardwareReset, 67
- IsTriggerMatching, 67
- nchannels, 79
- outBufTDCErr, 79
- outBufTDCHeadTrail, 79
- outBufTDCTTT, 79
- pair\_lead\_res, 79
- pair\_width\_res, 79
- ReadDetection, 68
- ReadFIFOSize, 68
- ReadGlobalOffset, 69
- ReadRCAdjust, 69
- ReadRegister, 70
- ReadResolution, 70
- ReadTrigConf, 71
- SetAcquisitionMode, 71
- SetBLTEventNumberRegister, 71
- SetContinuousStorage, 72
- SetCtlRegister, 72
- SetDetection, 72
- SetETTT, 73
- SetFIFOSize, 73
- SetGlobalOffset, 73
- SetLSBTrailEdge, 74
- SetPairModeResolution, 74
- SetPol, 75
- SetRCAdjust, 75
- SetStatusRegister, 75
- SetTDCEncapsulation, 75
- SetTDCErrorMarks, 75
- SetTriggerMatching, 76
- SetVerboseLevel, 76
- SetWindowOffset, 76
- SetWindowWidth, 77
- SoftwareClear, 77
- SoftwareReset, 77
- TDCV1x90, 63
- trailEdge\_res, 79
- WaitMicro, 78
- WriteRegister, 78
- VME::TDCV1x90Opcodes, 14
  - AUTOLOAD\_DEF\_CONFI, 16
  - AUTOLOAD\_USER\_CONF, 16
  - CLEAR\_KEEP\_TOKEN, 16
  - CONT\_STOR, 16
  - DIS\_ALL\_CHANNEL, 16
  - DIS\_CHANNEL, 16
  - DIS\_ERROR\_BYPASS, 16
  - DIS\_ERROR\_MARK, 16
  - DIS\_HEAD\_TRAILER, 16
  - DIS\_SUB\_TRG, 16
  - EN\_ALL\_CHANNEL, 16
  - EN\_CHANNEL, 16
  - EN\_ERROR\_BYPASS, 16
  - EN\_ERROR\_MARK, 16
  - EN\_HEAD\_TRAILER, 16
  - EN\_SUB\_TRG, 16
  - LOAD\_DEF\_CONFIG, 16
  - LOAD\_USER\_CONFIG, 16
  - READ\_ACQ\_MOD, 16
  - READ\_DEAD\_TIME, 16
  - READ\_DETECTION, 16
  - READ\_EN\_PATTERN, 16
  - READ\_EN\_PATTERN32, 16
  - READ\_ERROR\_TYPES, 16
  - READ\_EVENT\_SIZE, 16
  - READ\_FIFO\_SIZE, 16
  - READ\_GLOB\_OFFS, 16
  - READ\_HEAD\_TRAILER, 16
  - READ\_RC\_ADJ, 17
  - READ\_RES, 17
  - READ\_TRG\_CONF, 17
  - SAVE\_RC\_ADJ, 17
  - SAVE\_USER\_CONFIG, 17
  - SET\_DEAD\_TIME, 17
  - SET\_DETECTION, 17
  - SET\_ERROR\_TYPES, 17
  - SET\_EVENT\_SIZE, 17
  - SET\_FIFO\_SIZE, 17
  - SET\_GLOB\_OFFS, 17
  - SET\_KEEP\_TOKEN, 17
  - SET\_PAIR\_RES, 17
  - SET\_RC\_ADJ, 17
  - SET\_REJ\_MARGIN, 17
  - SET\_SW\_MARGIN, 17
  - SET\_TR\_LEAD\_LSB, 17
  - SET\_WIN\_OFFS, 17
  - SET\_WIN\_WIDTH, 17
  - TRG\_MATCH, 17
  - WRITE\_EN\_PATTERN, 17
  - WRITE\_EN\_PATTERN32, 17
- VME::glob\_offs, 30
  - coarse, 30
  - fine, 30
- VME::trailEdge\_t, 79
  - ettt, 79
  - event\_count, 79
  - leading, 80
  - total\_hits, 80
  - trailing, 80
- WEBSOCKET\_CLIENT
  - Socket communication objects, 9
- WIN\_OFFSET
  - VME, 14
- WRITE\_EN\_PATTERN

VME::TDCV1x90Opcodes, [17](#)  
WRITE\_EN\_PATTERN32  
VME::TDCV1x90Opcodes, [17](#)  
WRITE\_OK  
VME, [13](#)  
WaitMicro  
VME::TDCV1x90, [78](#)  
WriteRegister  
VME::TDCV1x90, [78](#)