2015 Test beam Run Control

Generated by Doxygen 1.8.9.1

Wed Jun 17 2015 18:57:43

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

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 BridgeType { CAEN_V1718, CAEN_V2718 }

  Compatible bridge types.
- 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, OTRAILING = 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↩
  AM_ENABLE = 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::BridgeType

Compatible bridge types.

Enumerator

CAEN_V1718

CAEN_V2718

6.1.2.3 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

6.1.2.4 enum VME::det_mode

Enumerator

PAIR
OTRAILING
OLEADING
TRAILEAD

6.1.2.5 enum VME::micro_handshake

Enumerator

WRITE_OK Is the TDC ready for writing?
READ_OK Is the TDC ready for reading?

6.1.2.6 enum VME::mod_reg

Enumerator

Control
Status
InterruptLevel
InterruptVector
GeoAddress
MCSTBase
MCSTControl
ModuleReset
kSoftwareClear
EventCounter
EventStored
BLTEventNumber
FirmwareRev
Micro
MicroHandshake
EventFIFO
EventFIFOStoredRegister
EventFIFOStatusRegister
ROMOui2
ROMOui1
ROMOui0
ROMBoard2
ROMBoard1
ROMBoard0
ROMRevis3
ROMRevis2
ROMRevis1
ROMRevis0
ROMSerNum1
ROMSerNum0

6.1.2.7 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

6.1.2.8 enum VME::trailead_edge_lsb

Enumerator

r800ps

r200ps

r100ps

r25ps

6.1.2.9 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_ADJUST_CH (0x5200)
- Opcode READ_ADJUST_CH (0x5200)
- Opcode SET_RC_ADJ (0x5400)
- Opcode READ_RC_ADJ (0x5500)
- Opcode SAVE_RC_ADJ (0x5600)
- Opcode READ_TDC_ID (0x6000)
- Opcode READ_MICRO_REV (0x6100)
- Opcode RESET_DLL_PLL (0x6200)
- Opcode WRITE_SETUP_REG (0x7000)
- Opcode READ_SETUP_REG (0x7100)
- Opcode UPDATE_SETUP_REG (0x7200)
- Opcode DEFAULT_SETUP_REG (0x7300)

- Opcode READ_ERROR_STATUS (0x7400)
- Opcode READ_DLL_LOCK (0x7500)
- Opcode READ_STATUS_STREAM (0x7600)
- Opcode UPDATE_SETUP_TDC (0x7700)
- Opcode WRITE_EEPROM (0xc000)
- Opcode READ_EEPROM (0xc100)
- Opcode REV_DATE_MICRO_FW (0xc200)
- Opcode WRITE_SPARE (0xc300)
- Opcode READ_SPARE (0xc400)
- Opcode ENABLE_TEST_MODE (0xc500)
- Opcode DISABLE_TEST_MODE (0xc600)
- Opcode SET_TDC_TSET_OUTPUT (0xc700)
- Opcode SET_DLL_CLOCK (0xc800)
- Opcode READ_SETUP_SCANPATH (0xc900)

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::DEFAULT_SETUP_REG (0x7300)

6.2.1.6 Opcode VME::TDCV1x90Opcodes::DIS_ALL_CHANNEL (0x4300)

6.2.1.7 Opcode VME::TDCV1x90Opcodes::DIS_CHANNEL (0x4100)

6.2.1.8 Opcode VME::TDCV1x90Opcodes::DIS_ERROR_BYPASS (0x3800)

6.2.1.9 Opcode VME::TDCV1x90Opcodes::DIS_ERROR_MARK (0x3600)

6.2.1.10 Opcode VME::TDCV1x90Opcodes::DIS_HEAD_TRAILER (0x3100)

6.2.1.11 Opcode VME::TDCV1x90Opcodes::DIS_SUB_TRG (0x1500)

6.2.1.12 Opcode VME::TDCV1x90Opcodes::DISABLE_TEST_MODE (0xc600)

6.2.1.13 Opcode VME::TDCV1x90Opcodes::EN_ALL_CHANNEL (0x4200)

6.2.1.14 Opcode VME::TDCV1x90Opcodes::EN_CHANNEL (0x4000)

6.2.1.15 Opcode VME::TDCV1x90Opcodes::EN_ERROR_BYPASS (0x3700)

6.2.1.16 Opcode VME::TDCV1x90Opcodes::EN_ERROR_MARK (0x3500)

6.2.1.17 Opcode VME::TDCV1x90Opcodes::EN_HEAD_TRAILER (0x3000)

6.2.1.18 Opcode VME::TDCV1x90Opcodes::EN_SUB_TRG (0x1400)

6.2.1.19 Opcode VME::TDCV1x90Opcodes::ENABLE_TEST_MODE (0xc500)

6.2.1.20 Opcode VME::TDCV1x90Opcodes::LOAD_DEF_CONFIG (0x0500)

6.2.1.21 Opcode VME::TDCV1x90Opcodes::LOAD_USER_CONFIG (0x0700)

6.2.1.22 Opcode VME::TDCV1x90Opcodes::READ_ACQ_MOD (0x0200)

6.2.1.23 Opcode VME::TDCV1x90Opcodes::READ_ADJUST_CH (0x5200)

6.2.1.24 Opcode VME::TDCV1x90Opcodes::READ_DEAD_TIME (0x2900)

6.2.1.25 Opcode VME::TDCV1x90Opcodes::READ_DETECTION (0x2300)

6.2.1.26 Opcode VME::TDCV1x90Opcodes::READ_DLL_LOCK (0x7500)

6.2.1.27 Opcode VME::TDCV1x90Opcodes::READ_EEPROM (0xc100)

6.2.1.28 Opcode VME::TDCV1x90Opcodes::READ_EN_PATTERN (0x4500)

6.2.1.29 Opcode VME::TDCV1x90Opcodes::READ_EN_PATTERN32 (0x4700)

6.2.1.30 Opcode VME::TDCV1x90Opcodes::READ_ERROR_STATUS (0x7400)

6.2.1.31 Opcode VME::TDCV1x90Opcodes::READ_ERROR_TYPES (0x3a00)

6.2.1.32 Opcode VME::TDCV1x90Opcodes::READ_EVENT_SIZE (0x3400)

6.2.1.33 Opcode VME::TDCV1x90Opcodes::READ_FIFO_SIZE (0x3c00)

6.2.1.34 Opcode VME::TDCV1x90Opcodes::READ_GLOB_OFFS (0x5100)

6.2.1.35 Opcode VME::TDCV1x90Opcodes::READ_HEAD_TRAILER (0x3200)

6.2.1.36 Opcode VME::TDCV1x90Opcodes::READ_MICRO_REV (0x6100)

6.2.1.37 Opcode VME::TDCV1x90Opcodes::READ_RC_ADJ (0x5500)

6.2.1.38 Opcode VME::TDCV1x90Opcodes::READ_RES (0x2600)

6.2.1.39 Opcode VME::TDCV1x90Opcodes::READ_SETUP_REG (0x7100)

6.2.1.40 Opcode VME::TDCV1x90Opcodes::READ_SETUP_SCANPATH (0xc900)

6.2.1.41 Opcode VME::TDCV1x90Opcodes::READ_SPARE (0xc400)

6.2.1.42 Opcode VME::TDCV1x90Opcodes::READ_STATUS_STREAM (0x7600)

6.2.1.43 Opcode VME::TDCV1x90Opcodes::READ_TDC_ID (0x6000)

6.2.1.44 Opcode VME::TDCV1x90Opcodes::READ_TRG_CONF (0x1600)

6.2.1.45 Opcode VME::TDCV1x90Opcodes::RESET_DLL_PLL (0x6200)

6.2.1.46 Opcode VME::TDCV1x90Opcodes::REV_DATE_MICRO_FW (0xc200)

6.2.1.47 Opcode VME::TDCV1x90Opcodes::SAVE_RC_ADJ (0x5600)

6.2.1.48 Opcode VME::TDCV1x90Opcodes::SAVE_USER_CONFIG (0x0600)

6.2.1.49 Opcode VME::TDCV1x90Opcodes::SET_ADJUST_CH (0x5200)

6.2.1.50 Opcode VME::TDCV1x90Opcodes::SET_DEAD_TIME (0x2800)

6.2.1.51 Opcode VME::TDCV1x90Opcodes::SET_DETECTION (0x2200)

6.2.1.52 Opcode VME::TDCV1x90Opcodes::SET_DLL_CLOCK (0xc800)

6.2.1.53 Opcode VME::TDCV1x90Opcodes::SET_ERROR_TYPES (0x3900)

6.2.1.54 Opcode VME::TDCV1x90Opcodes::SET_EVENT_SIZE (0x3300)

6.2.1.55 Opcode VME::TDCV1x90Opcodes::SET_FIFO_SIZE (0x3b00)

6.2.1.56 Opcode VME::TDCV1x90Opcodes::SET_GLOB_OFFS (0x5000)

6.2.1.57 Opcode VME::TDCV1x90Opcodes::SET_KEEP_TOKEN (0x0300)

6.2.1.58 Opcode VME::TDCV1x90Opcodes::SET_PAIR_RES (0x2500)

6.2.1.59 Opcode VME::TDCV1x90Opcodes::SET_RC_ADJ (0x5400)

6.2.1.60 Opcode VME::TDCV1x90Opcodes::SET_REJ_MARGIN (0x1300)

6.2.1.61 Opcode VME::TDCV1x90Opcodes::SET_SW_MARGIN (0x1200)

6.2.1.62 Opcode VME::TDCV1x90Opcodes::SET_TDC_TSET_OUTPUT (0xc700)

6.2.1.63 Opcode VME::TDCV1x90Opcodes::SET_TR_LEAD_LSB (0x2400)

6.2.1.64 Opcode VME::TDCV1x90Opcodes::SET_WIN_OFFS (0x1100)

6.2.1.65 Opcode VME::TDCV1x90Opcodes::SET_WIN_WIDTH (0x1000)

6.2.1.66 Opcode VME::TDCV1x90Opcodes::TRG_MATCH (0x0000)

6.2.1.67 Opcode VME::TDCV1x90Opcodes::UPDATE_SETUP_REG (0x7200)

6.2.1.68 Opcode VME::TDCV1x90Opcodes::UPDATE_SETUP_TDC (0x7700)

6.2.1.69 Opcode VME::TDCV1x90Opcodes::WRITE_EEPROM (0xc000)

6.2.1.70 Opcode VME::TDCV1x90Opcodes::WRITE_EN_PATTERN (0x4400)

6.2.1.71 Opcode VME::TDCV1x90Opcodes::WRITE_EN_PATTERN32 (0x4600)

6.2.1.72 Opcode VME::TDCV1x90Opcodes::WRITE_SETUP_REG (0x7000)

6.2.1.73 Opcode VME::TDCV1x90Opcodes::WRITE_SPARE (0xc300)

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, BridgeType type)

  Constructor.
- ~BridgeVx718 ()

  Destructor.
- int32_t GetHandle () const

  Gets bhandle.
- void CheckConfiguration () const
- 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
Bob Velghe bob.velghe@cern.ch

Date

Jun 2010

7.1.2 Constructor & Destructor Documentation

7.1.2.1 VME::BridgeVx718::BridgeVx718 (const char * *device*, BridgeType *type*)

Constructor.

Bridge class constructor

Parameters

in	*device*	Device identifier on the VME crate
in	*type*	Device type (1718/2718)

Here is the call graph for this function:

7.1.2.2 VME::BridgeVx718::~BridgeVx718 ( )

Destructor.

Bridge class destructor

7.1.3 Member Function Documentation

7.1.3.1 void VME::BridgeVx718::CheckConfiguration ( ) const

7.1.3.2 int32_t VME::BridgeVx718::GetHandle ( ) const [inline]

Gets bhandle.

Gives bhandle value

Returns

bhandle value

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

Set and read the input lines.

7.1.3.4 void VME::BridgeVx718::InputRead (CVInputSelect *input*)

7.1.3.5 void VME::BridgeVx718::OutputConf (CVOutputSelect *output*)

Set and control the output lines.

7.1.3.6 void VME::BridgeVx718::OutputOff (CVOutputSelect *output*)

7.1.3.7 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:

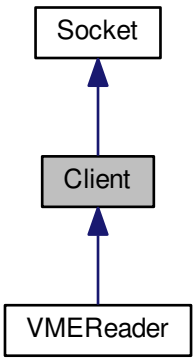
- include/VME_BridgeVx718.h
- src/VME_BridgeVx718.cpp

7.2 Client Class Reference

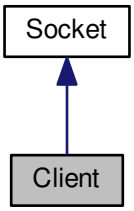
Base client object for the socket.

`#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.
- SocketMessage SendAndReceive (const SocketMessage &m, const MessageKey &a) const
- 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 fIsConnected

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

Date

24 Mar 2015

7.2.2 Constructor & Destructor Documentation

7.2.2.1 Client::Client ( ) [inline]

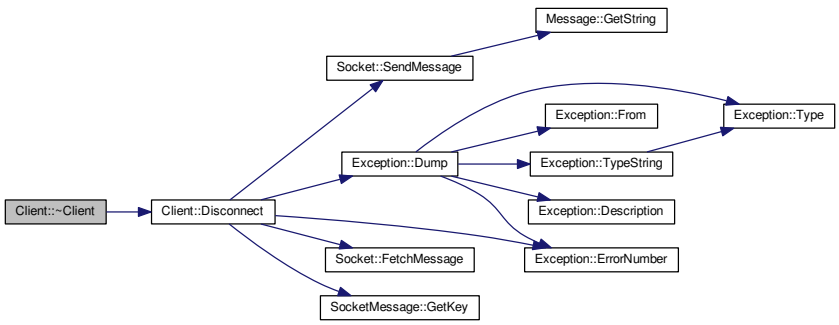
General void client constructor.

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:

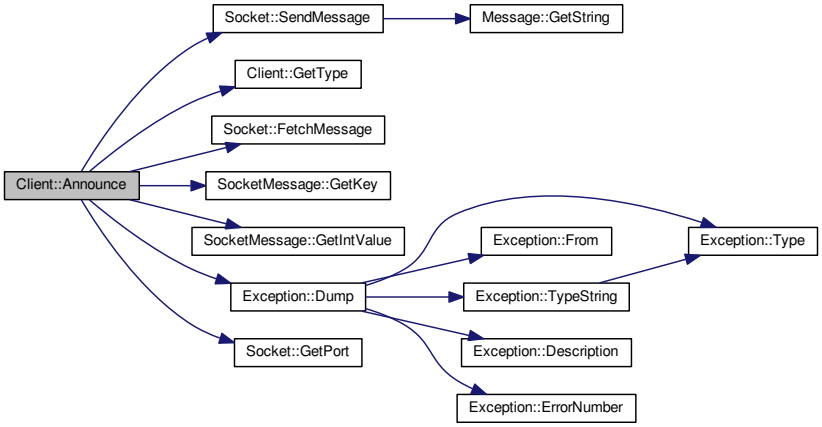


7.2.3 Member Function Documentation

7.2.3.1 void Client::Announce ( ) [private]

Announce our entry on the socket to its master.

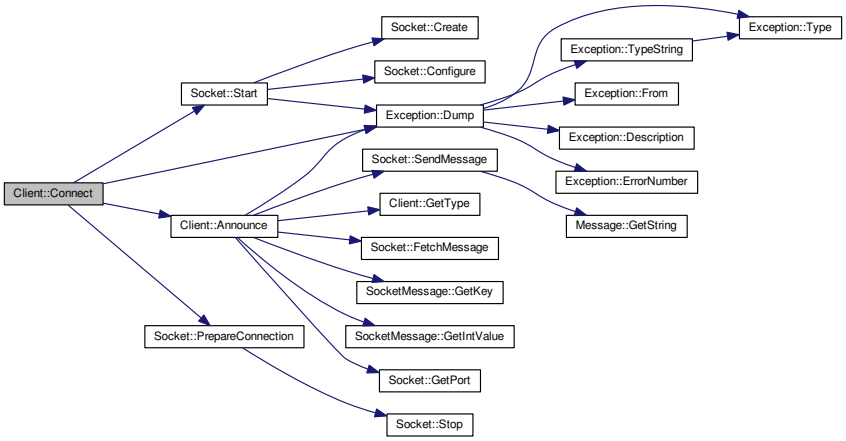
Here is the call graph for this function:



7.2.3.2 bool Client::Connect ( )

Bind this client to the socket.

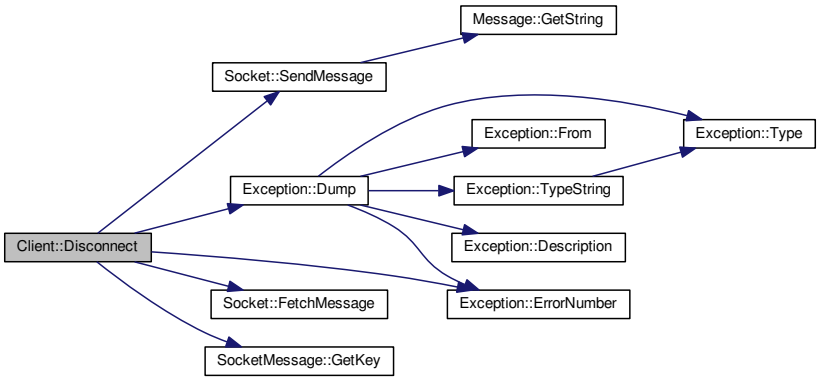
Here is the call graph for this function:



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.

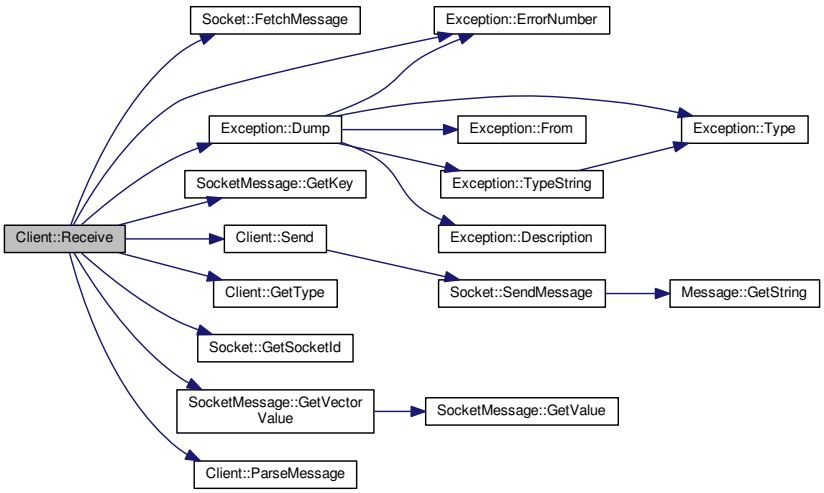
7.2.3.5 virtual void Client::ParseMessage (const SocketMessage & *m*) [inline], [virtual]

Parse a SocketMessage received from the master.

7.2.3.6 void Client::Receive ( )

Receive a socket message from the master.

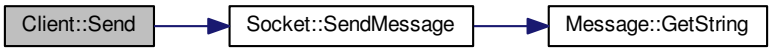
Here is the call graph for this function:



7.2.3.7 void Client::Send (const Message & *m*) const [inline]

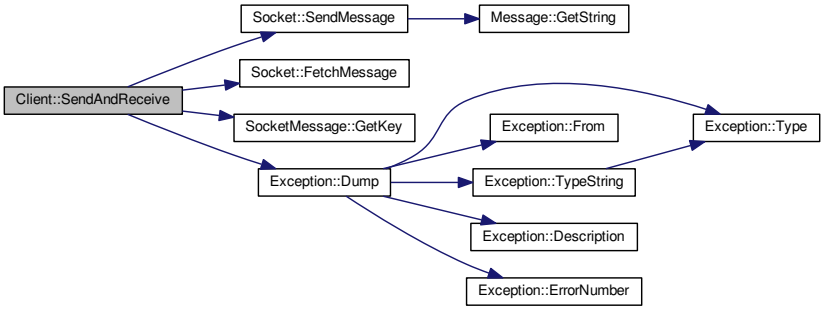
Send a message to the master through the socket.

Here is the call graph for this function:



7.2.3.8 SocketMessage Client::SendAndReceive (const SocketMessage & *m*, const MessageKey & *a*) const [inline]

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

Date

24 Mar 2015

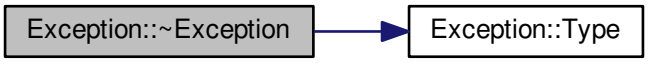
7.3.2 Constructor & Destructor Documentation

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:

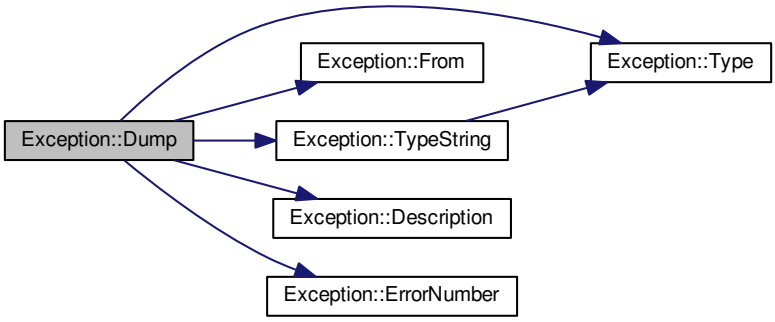


7.3.3 Member Function Documentation

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:



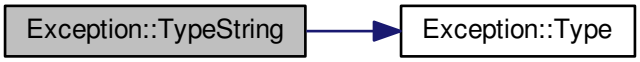
7.3.3.3 int Exception::ErrorNumber ( ) const [inline]

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

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

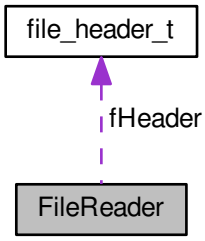
The documentation for this struct was generated from the following file:

- 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:

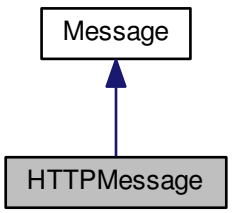
- include/VME_TDCV1x90.h

7.7 HTTPMessage Class Reference

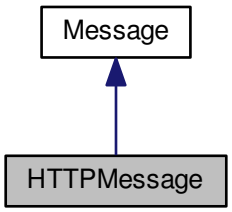
Message to be transmitted through a WebSocket protocol.

`#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

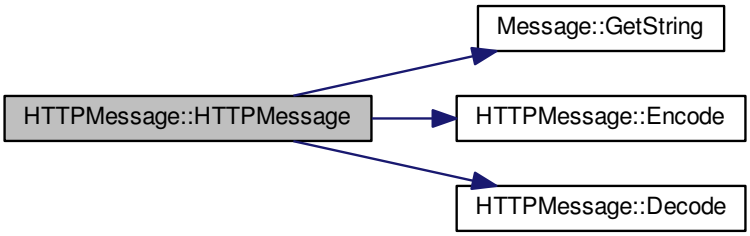
Date

1 Apr 2015

7.7.2 Constructor & Destructor Documentation

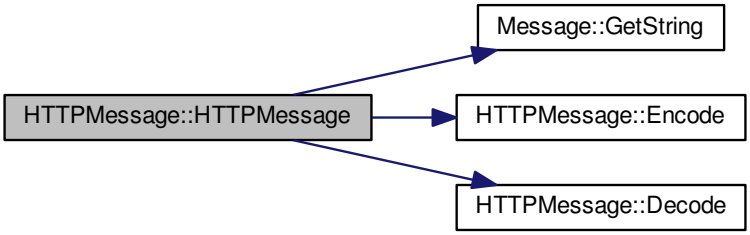
7.7.2.1 HTTPMessage::HTTPMessage (WebSocket * *ws*, Message *m*, MessageAction *a*) [inline]

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:

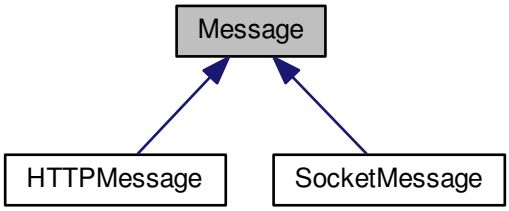
- include/HTTPMessage.h

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

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:

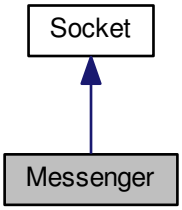
- include/Message.h

7.9 Messenger Class Reference

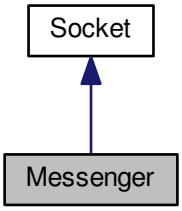
Base master object for the socket.

`#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.

- void StartAcquisition ()

  Start the data acquisition.

- void StopAcquisition ()
- 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
- pid_t fPID

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

Date

23 Mar 2015

7.9.2 Constructor & Destructor Documentation

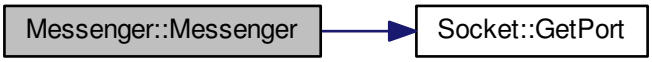
7.9.2.1 Messenger::Messenger ( )

Build a void master object or socket actor.

7.9.2.2 Messenger::Messenger (int *port*)

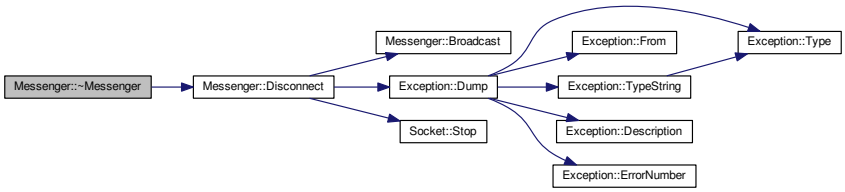
Build a master object to control the socket.

Here is the call graph for this function:



7.9.2.3 Messenger::~Messenger ( )

Here is the call graph for this function:



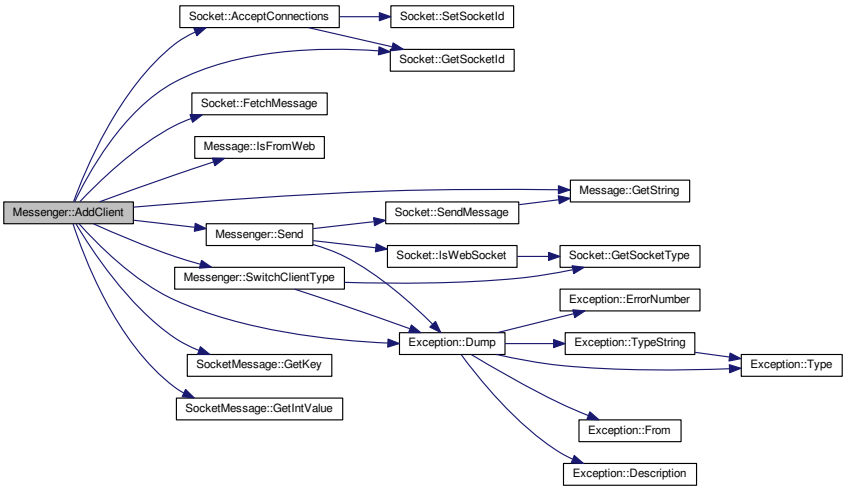
7.9.3 Member Function Documentation

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 & *m*) const

Emit a message to all clients connected through the socket.

Parameters

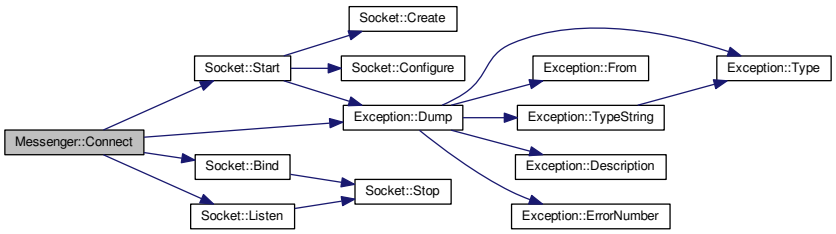
in	*m*	Message to transmit

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:

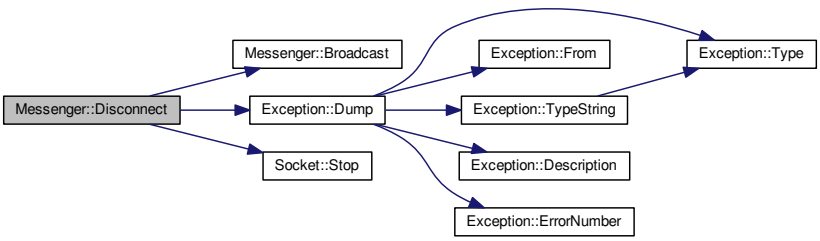


7.9.3.4 void Messenger::Disconnect ( )

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]

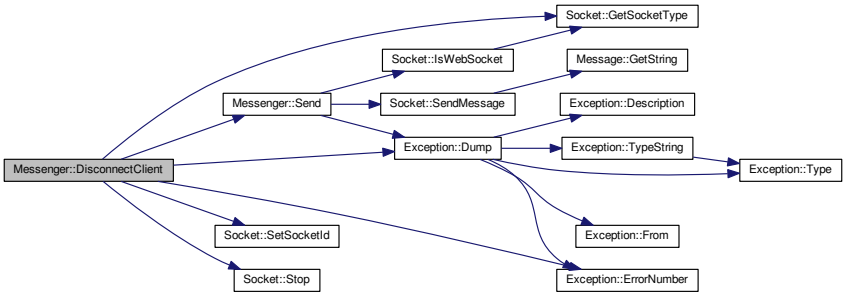
Disconnect a client.

Ask to a client to disconnect from this socket.

Parameters

in	*sid*	Unique identifier of the client to disconnect
in	*key*	Key to the message to transmit for disconnection
in	*force*	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.

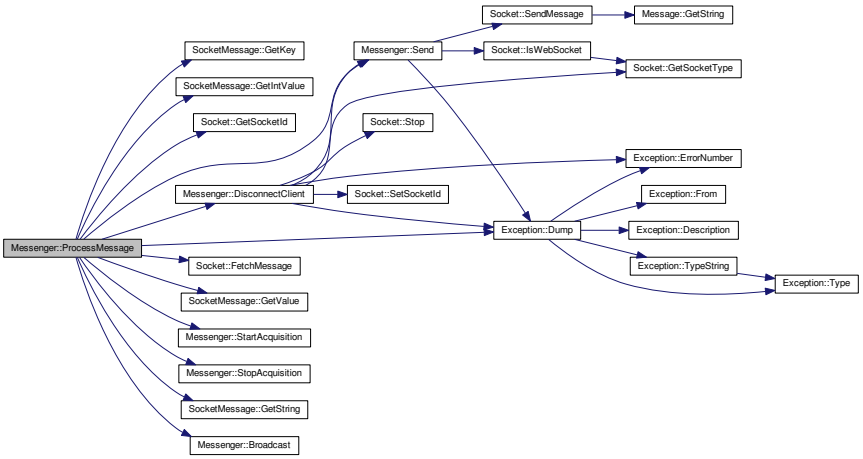
7.9.3.7 void Messenger::ProcessMessage (SocketMessage *m*, int *sid*) [private]

Process a message received from the socket.

Parameters

in	*Unique*	identifier of the client sending the message

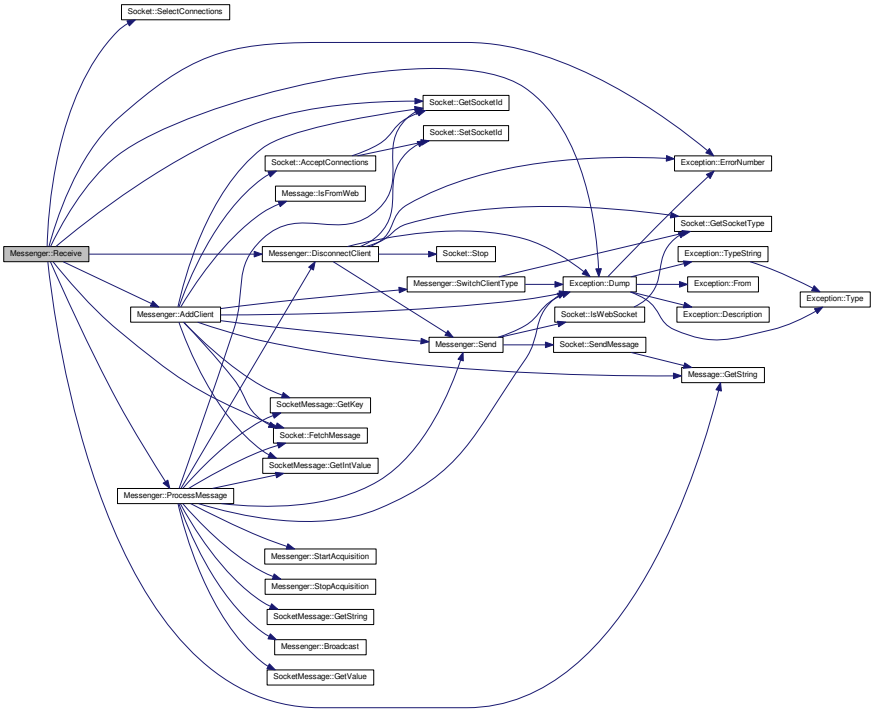
Here is the call graph for this function:



7.9.3.8 void Messenger::Receive ( )

Handle a message reception from a client.

Here is the call graph for this function:



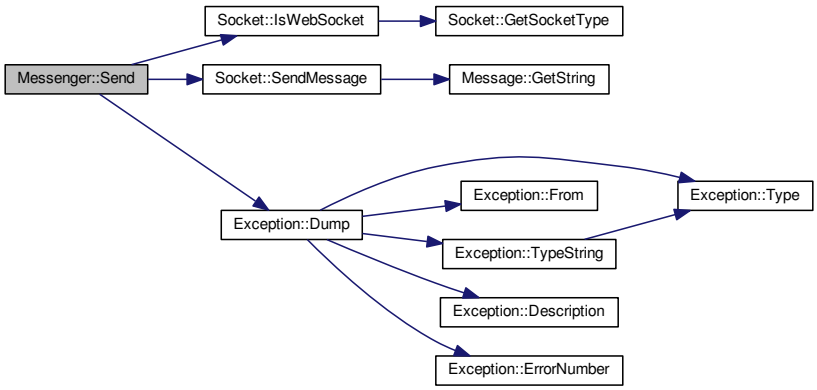
7.9.3.9 void Messenger::Send (const Message & *m,* int *sid*) const [inline]

Send any type of message to any client.

Parameters

in	*m*	Message to transmit
in	*sid*	Unique identifier of the client on this socket

Here is the call graph for this function:



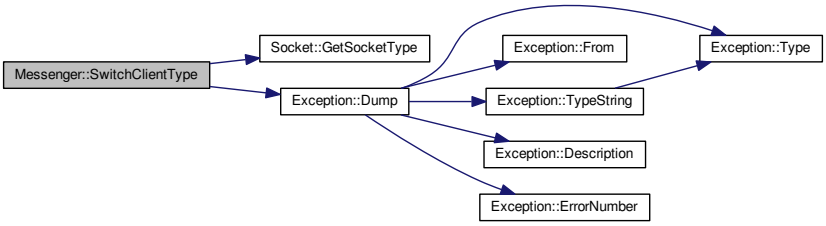
7.9.3.10 void Messenger::StartAcquisition ( )

Start the data acquisition.

7.9.3.11 void Messenger::StopAcquisition ( )

7.9.3.12 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 pid_t Messenger::fPID [private]

7.9.4.3 WebSocket* Messenger::fWS [private]

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

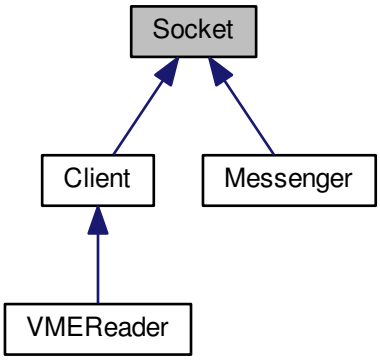
- include/Messenger.h
- src/Messenger.cpp

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

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*)

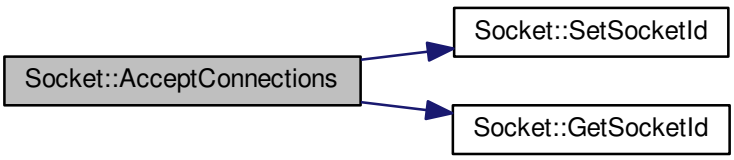
Accept connection from a client.

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

Parameters

in,out	*socket*	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]

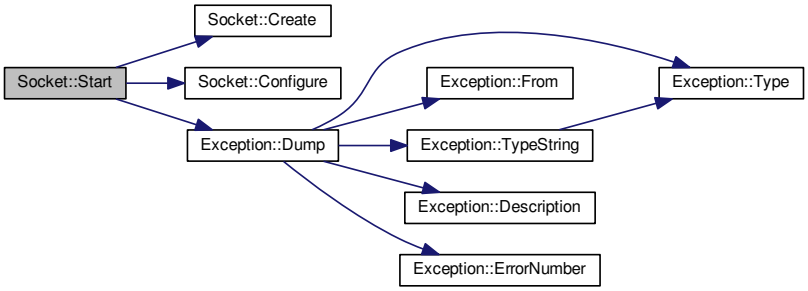
Start the socket.

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:

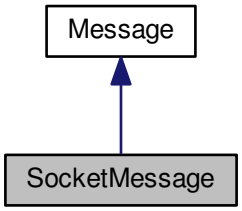
- include/Socket.h
- src/Socket.cpp

7.11 SocketMessage Class Reference

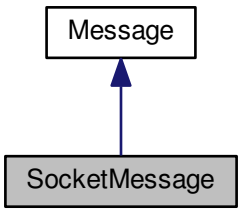
Socket-passed message type.

`#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

Date

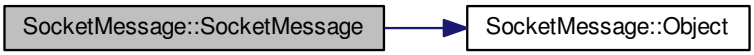
26 Mar 2015

7.11.2 Constructor & Destructor Documentation

7.11.2.1 SocketMessage::SocketMessage ( ) [inline]

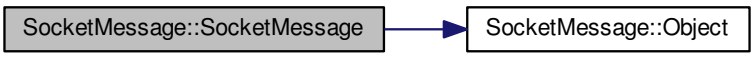
7.11.2.2 SocketMessage::SocketMessage (const Message & *msg*) [inline]

Here is the call graph for this function:



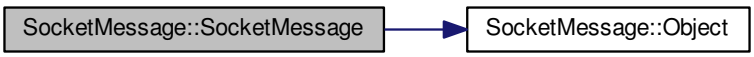
7.11.2.3 SocketMessage::SocketMessage (const char * *msg_s*) [inline]

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]

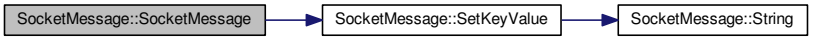
Construct a socket message out of a key and a string-type value.

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.

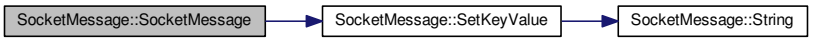
Here is the call graph for this function:



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.

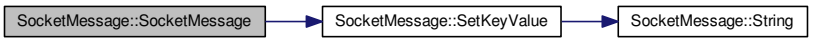
Here is the call graph for this function:



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.

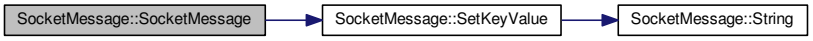
Here is the call graph for this function:



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]

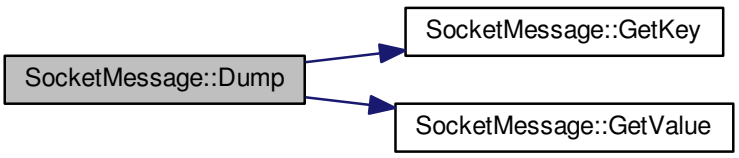
Construct a socket message out of a map of key/string-type value.

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.

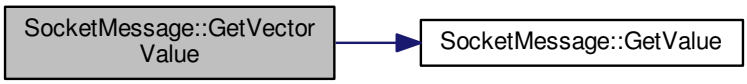
7.11.3.5 std::string SocketMessage::GetValue ( ) const [inline]

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:

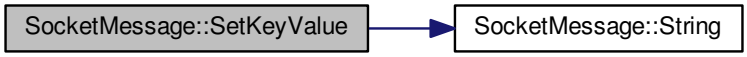


7.11.3.7 MessageMap SocketMessage::Object ( ) const [inline], [private]

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]

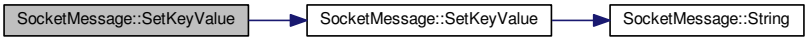
Send an integer-valued message.

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.

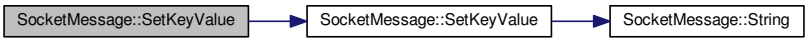
Here is the call graph for this function:



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 Dump () const
- 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.
- uint8_t GetStatus () const
- 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

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]

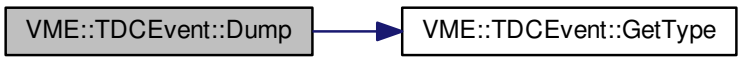
7.12.3.2 VME::TDCEvent::TDCEvent (const uint32_t & *word*) [inline]

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

7.12.4 Member Function Documentation

7.12.4.1 void VME::TDCEvent::Dump ( ) const [inline]

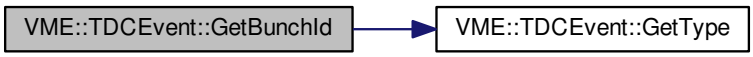
Here is the call graph for this function:



7.12.4.2 uint16_t VME::TDCEvent::GetBunchId ( ) const [inline]

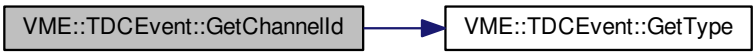
Bunch identifier of trigger (or trigger time tag)

Here is the call graph for this function:



7.12.4.3 uint8_t VME::TDCEvent::GetChannelId ( ) const [inline]

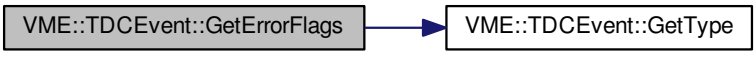
Here is the call graph for this function:



7.12.4.4 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:



7.12.4.5 uint32_t VME::TDCEvent::GetETTT ( ) const [inline]

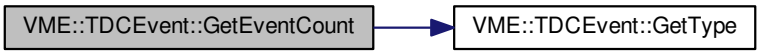
Extended trigger time tag.

Here is the call graph for this function:

7.12.4.6 uint32_t VME::TDCEvent::GetEventCount ( ) const [inline]

Total number of events.

Here is the call graph for this function:



7.12.4.7 uint16_t VME::TDCEvent::GetEventId ( ) const [inline]

Event identifier from event counter.

Here is the call graph for this function:

7.12.4.8 uint8_t VME::TDCEvent::GetGeo ( ) const [inline]

Here is the call graph for this function:

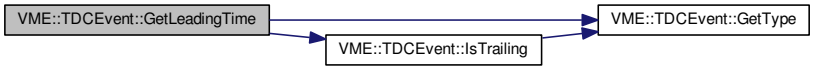
7.12.4.9 uint32_t VME::TDCEvent::GetLeadingTime (bool *pair* = false) const [inline]

Leading edge measurement in programmed time resolution.

Parameters

in	*pair*	Are we dealing with a pair measurement?

Here is the call graph for this function:



7.12.4.10 uint8_t VME::TDCEvent::GetStatus ( ) const [inline]

Here is the call graph for this function:

7.12.4.11 uint8_t VME::TDCEvent::GetTDCId ( ) const [inline]

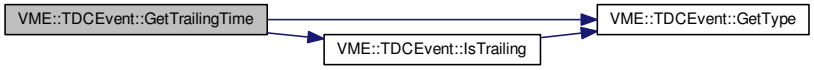
Programmed identifier of master TDC providing the event.

Here is the call graph for this function:

7.12.4.12 uint32_t VME::TDCEvent::GetTrailingTime ( ) const [inline]

Trailing edge measurement in programmed time resolution.

Here is the call graph for this function:



7.12.4.13 EventType VME::TDCEvent::GetType ( ) const [inline]

Type of packet read out from the TDC.

7.12.4.14 uint8_t VME::TDCEvent::GetWidth ( ) const [inline]

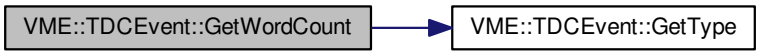
Width of pulse in programmed time resolution.

Here is the call graph for this function:

7.12.4.15 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:



7.12.4.16 bool VME::TDCEvent::IsTrailing ( ) const [inline]

Are we dealing with a trailing or a leading measurement?

Here is the call graph for this function:

7.12.4.17 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=1)
- uint32_t GetModel ()
- uint32_t GetOUI ()
- uint32_t GetSerialNumber ()
- void CheckConfiguration ()
- void EnableChannel (short)
- void DisableChannel (short)
- void SetPoI (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 FetchEvents ()
- 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 outBufTDCHeadTrail
- 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

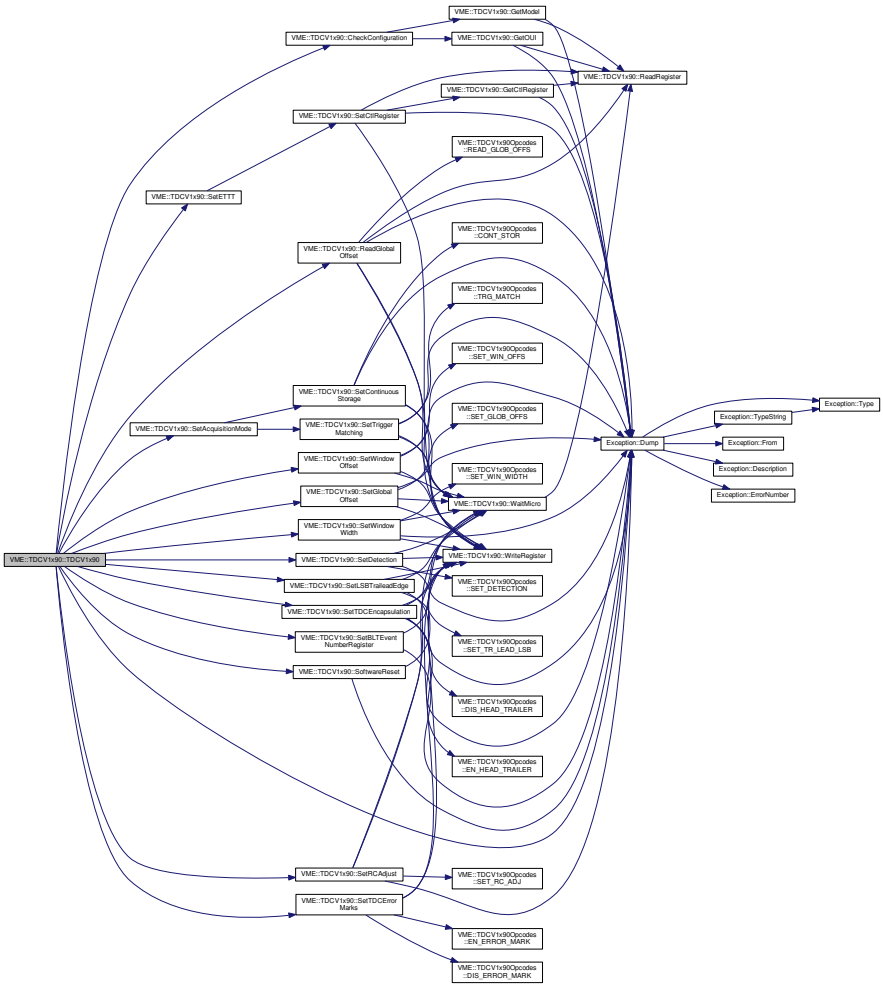
Date

Jun 2010

7.13.2 Constructor & Destructor Documentation

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

Here is the call graph for this function:



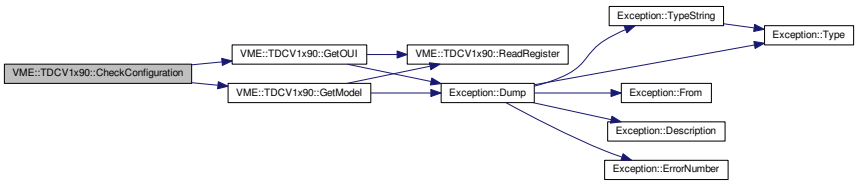
7.13.2.2 VME::TDCV1x90::∼TDCV1x90 ( )

7.13.3 Member Function Documentation

7.13.3.1 void VME::TDCV1x90::abort ( )

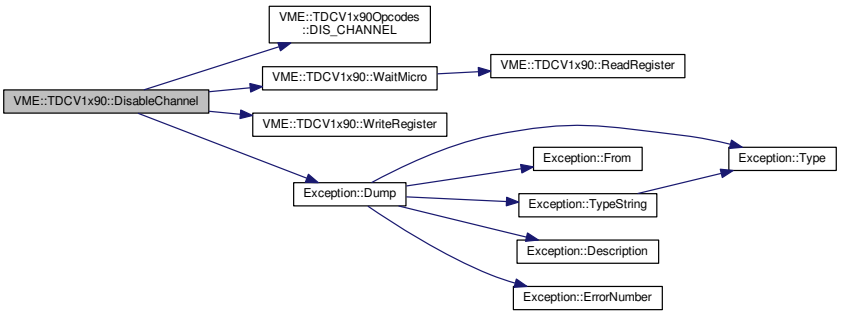
7.13.3.2 void VME::TDCV1x90::CheckConfiguration ( )

Here is the call graph for this function:



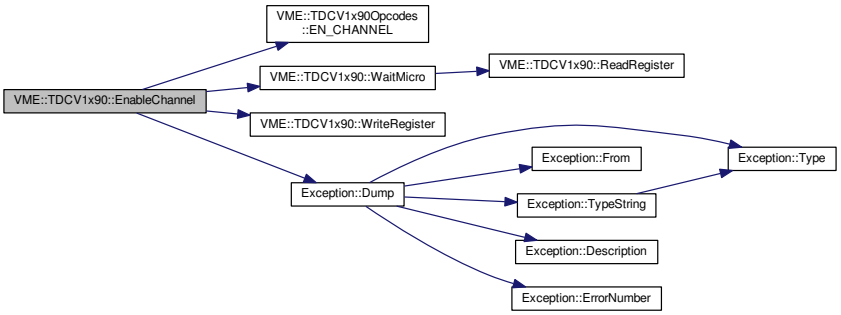
7.13.3.3 void VME::TDCV1x90::DisableChannel (short *channel_id*)

Here is the call graph for this function:



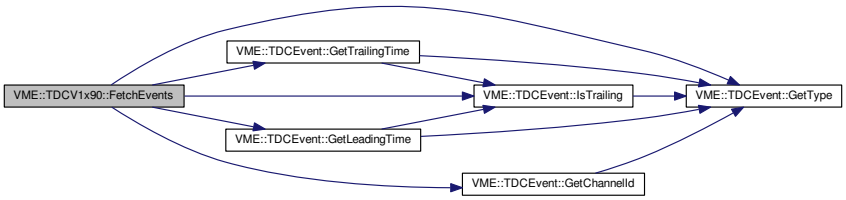
7.13.3.4 void VME::TDCV1x90::EnableChannel (short *channel_id*)

Here is the call graph for this function:



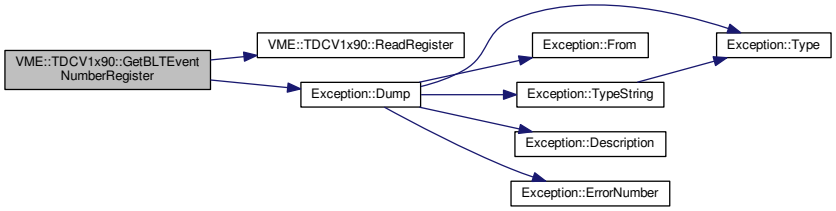
7.13.3.5 TDCEventCollection VME::TDCV1x90::FetchEvents ( )

Here is the call graph for this function:



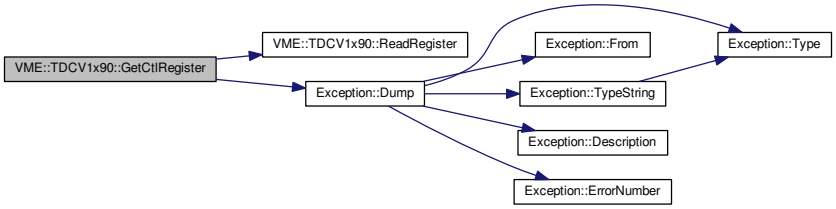
7.13.3.6 uint16_t VME::TDCV1x90::GetBLTEventNumberRegister ( )

Here is the call graph for this function:



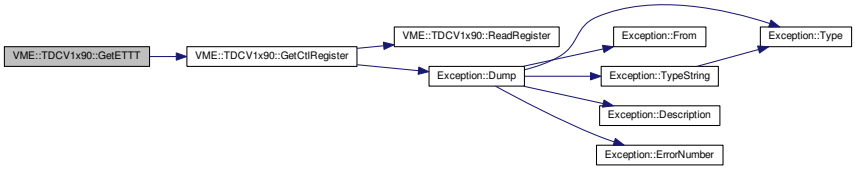
7.13.3.7 bool VME::TDCV1x90::GetCtlRegister (ctl_reg *bit*)

Here is the call graph for this function:



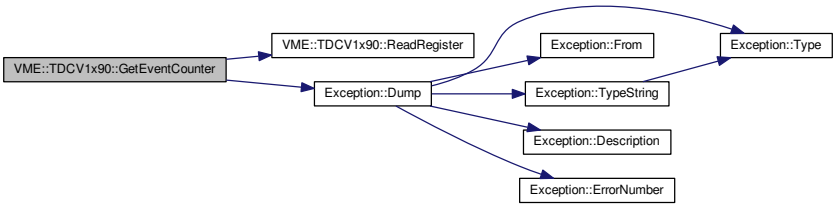
7.13.3.8 bool VME::TDCV1x90::GetETTT ( )

Here is the call graph for this function:



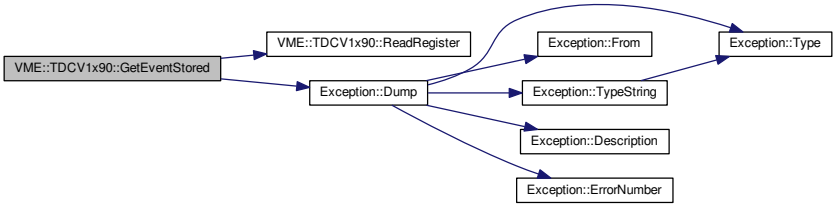
7.13.3.9 uint32_t VME::TDCV1x90::GetEventCounter ( )

Here is the call graph for this function:



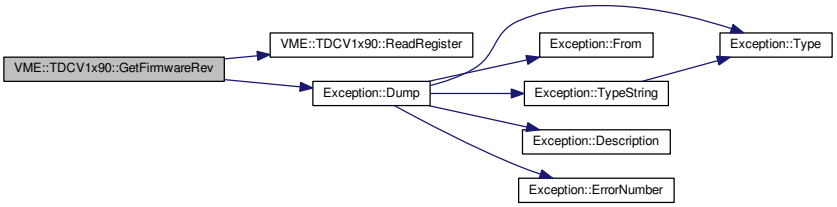
7.13.3.10 uint16_t VME::TDCV1x90::GetEventStored ( )

Here is the call graph for this function:



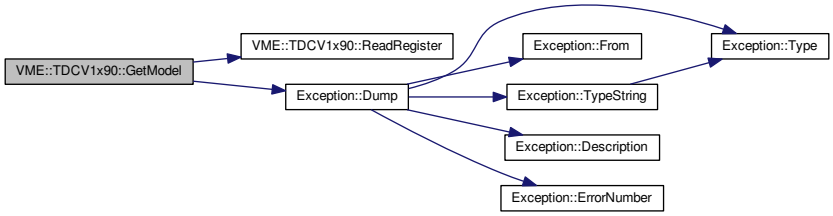
7.13.3.11 void VME::TDCV1x90::GetFirmwareRev ( )

Here is the call graph for this function:



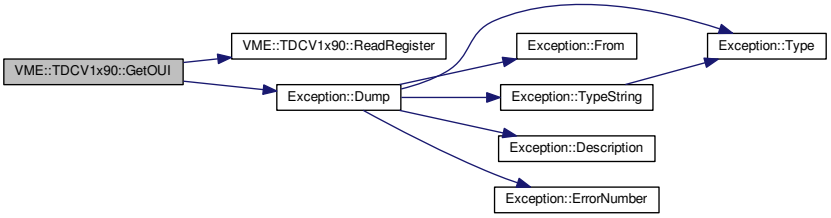
7.13.3.12 uint32_t VME::TDCV1x90::GetModel ( )

Here is the call graph for this function:



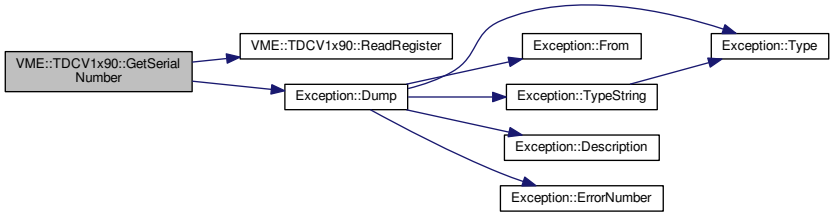
7.13.3.13 uint32_t VME::TDCV1x90::GetOUI ( )

Here is the call graph for this function:



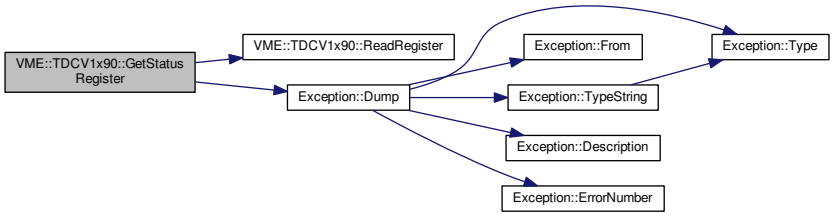
7.13.3.14 uint32_t VME::TDCV1x90::GetSerialNumber ( )

Here is the call graph for this function:



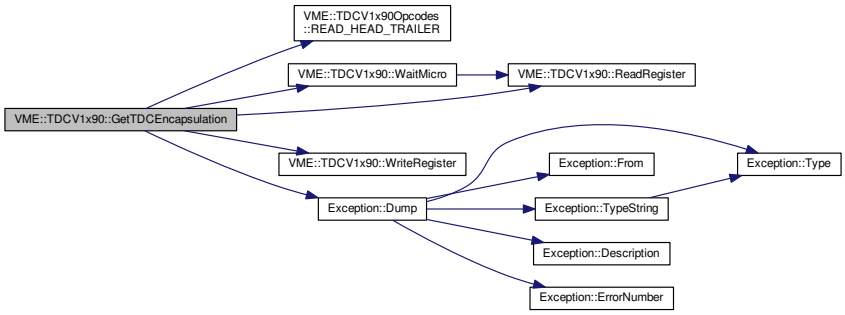
7.13.3.15 bool VME::TDCV1x90::GetStatusRegister (stat_reg *bit*)

Here is the call graph for this function:



7.13.3.16 bool VME::TDCV1x90::GetTDCEncapsulation ( )

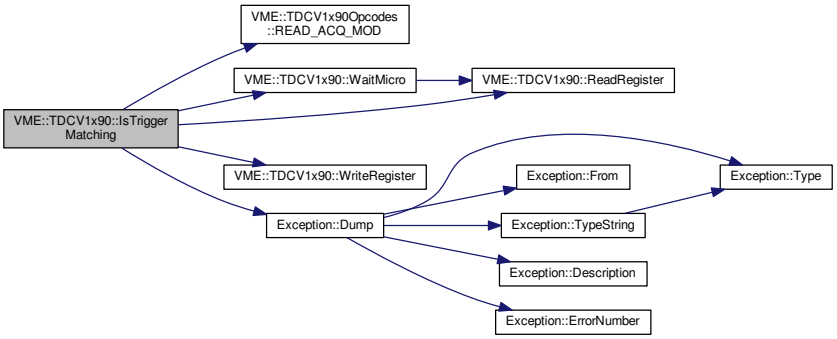
Here is the call graph for this function:



7.13.3.17 bool VME::TDCV1x90::HardwareReset ( )

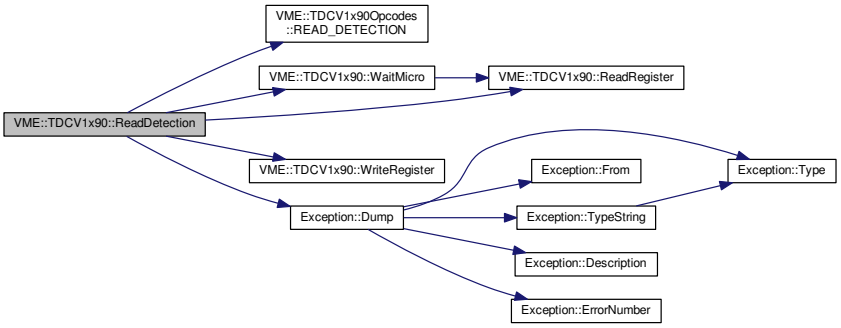
7.13.3.18 bool VME::TDCV1x90::IsTriggerMatching ( )

Here is the call graph for this function:



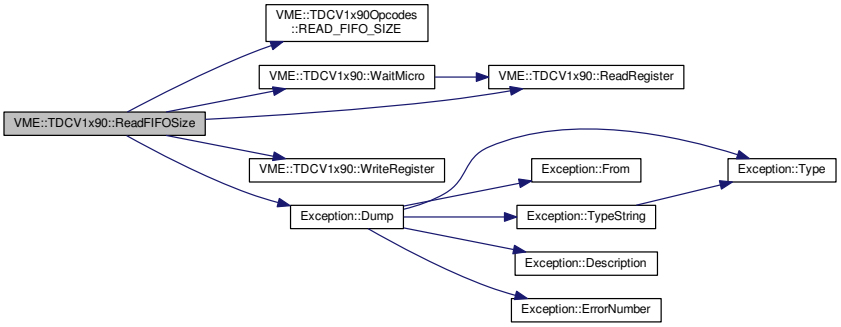
7.13.3.19 det_mode VME::TDCV1x90::ReadDetection ( )

Here is the call graph for this function:



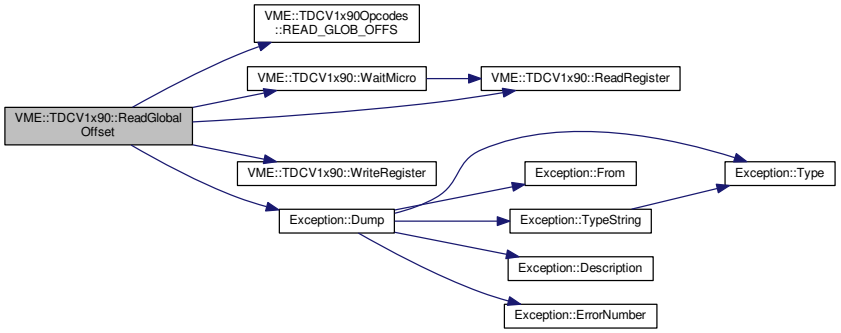
7.13.3.20 void VME::TDCV1x90::ReadFIFOSize ( )

Here is the call graph for this function:



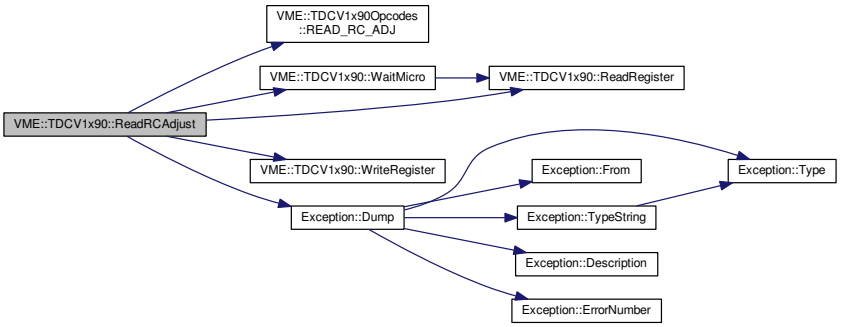
7.13.3.21 glob_offs VME::TDCV1x90::ReadGlobalOffset ( )

Here is the call graph for this function:



7.13.3.22 uint16_t VME::TDCV1x90::ReadRCAdjust (int *tdc*)

Here is the call graph for this function:



7.13.3.23 void VME::TDCV1x90::ReadRegister (mod_reg *addr,* uint16_t * *data*)

Read on register.

Read a 16-bit word in the register

Parameters

in	*addr*	register
out	*data*	word

7.13.3.24 void VME::TDCV1x90::ReadRegister (mod_reg *addr,* uint32_t * *data*)

Read on register.

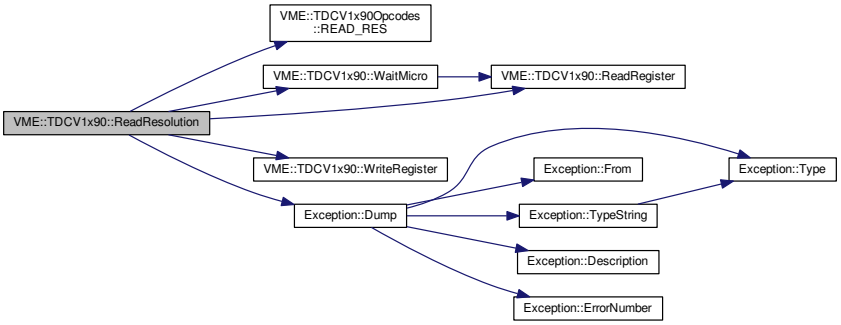
Read a 32-bit word in the register

Parameters

in	*addr*	register
out	*data*	word

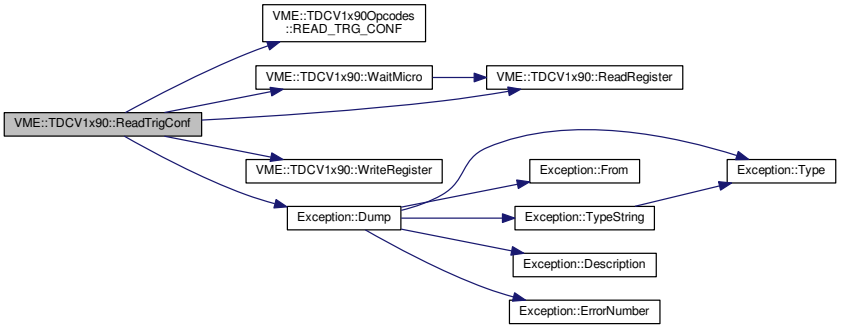
7.13.3.25 void VME::TDCV1x90::ReadResolution (det_mode *det*)

Here is the call graph for this function:



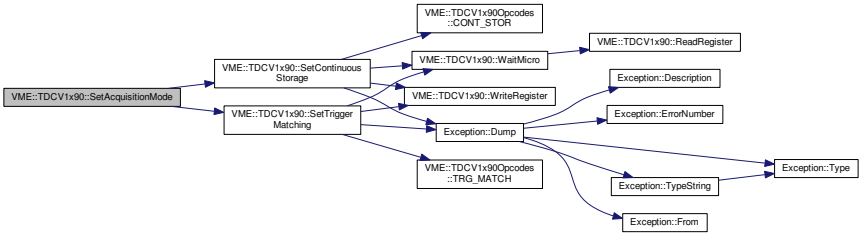
7.13.3.26 uint16_t VME::TDCV1x90::ReadTrigConf (trig_conf *type*)

Here is the call graph for this function:



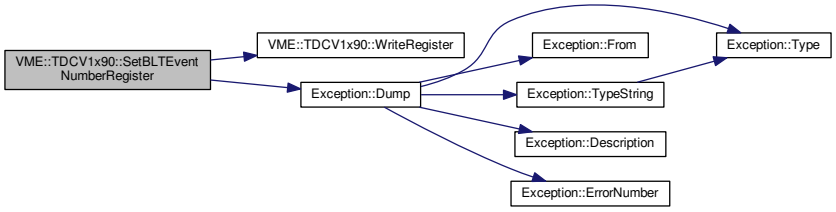
7.13.3.27 void VME::TDCV1x90::SetAcquisitionMode (acq_mode *mode*)

Here is the call graph for this function:



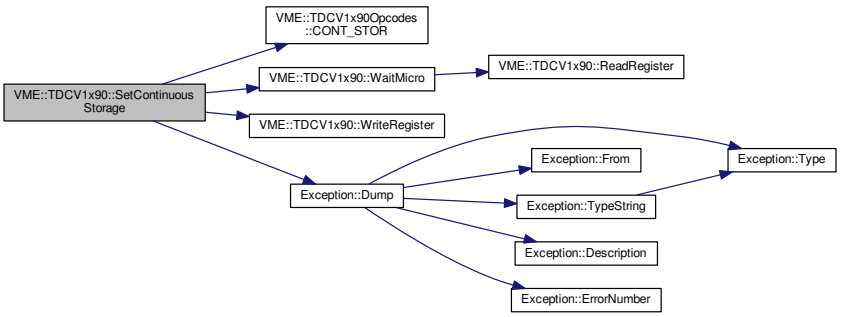
7.13.3.28 void VME::TDCV1x90::SetBLTEventNumberRegister (uint16_t *value*)

Here is the call graph for this function:



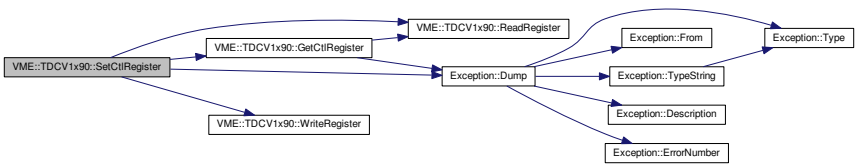
7.13.3.29 bool VME::TDCV1x90::SetContinuousStorage ( )

Here is the call graph for this function:



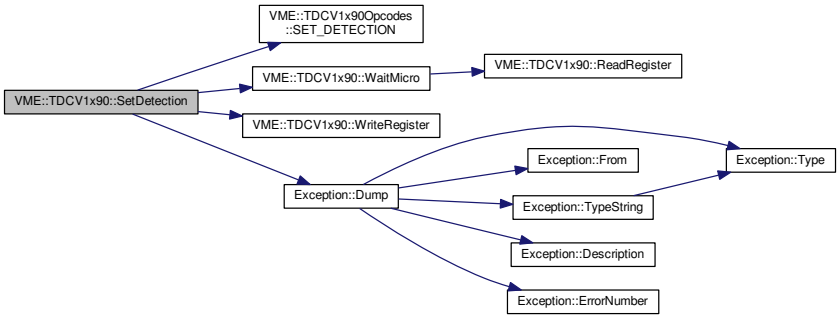
7.13.3.30 void VME::TDCV1x90::SetCtlRegister (ctl_reg *reg,* bool *value*)

Here is the call graph for this function:



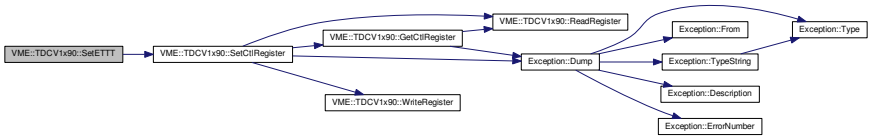
7.13.3.31 void VME::TDCV1x90::SetDetection (det_mode *mode*)

Here is the call graph for this function:



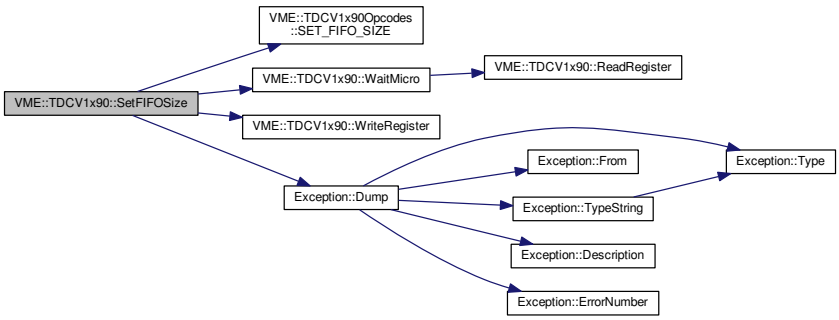
7.13.3.32 void VME::TDCV1x90::SetETTT (bool *mode*)

Here is the call graph for this function:



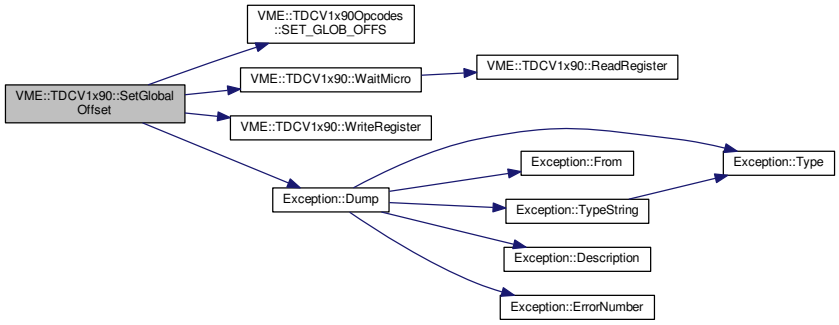
7.13.3.33 void VME::TDCV1x90::SetFIFOSize (uint16_t *size*)

Here is the call graph for this function:



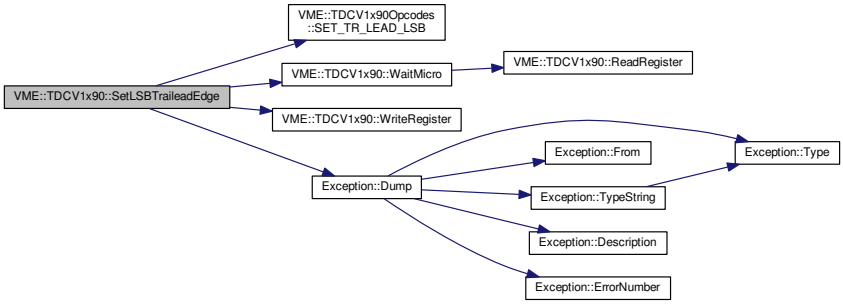
7.13.3.34 void VME::TDCV1x90::SetGlobalOffset (uint16_t *word1*, uint16_t *word2*)

Here is the call graph for this function:



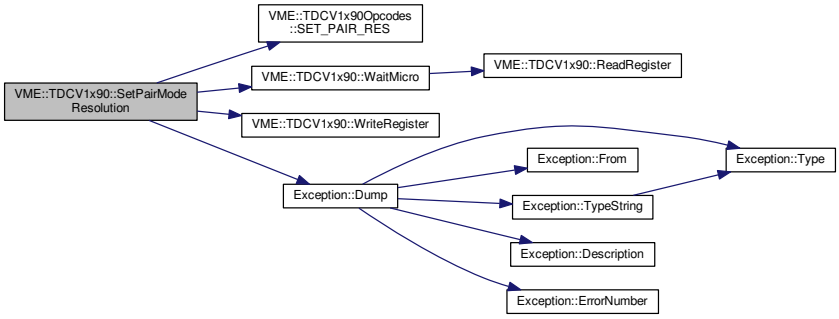
7.13.3.35 void VME::TDCV1x90::SetLSBTraileadEdge (trailead_edge_lsb *conf*)

Here is the call graph for this function:



7.13.3.36 void VME::TDCV1x90::SetPairModeResolution (int *lead_time_res*, int *pulse_width_res*)

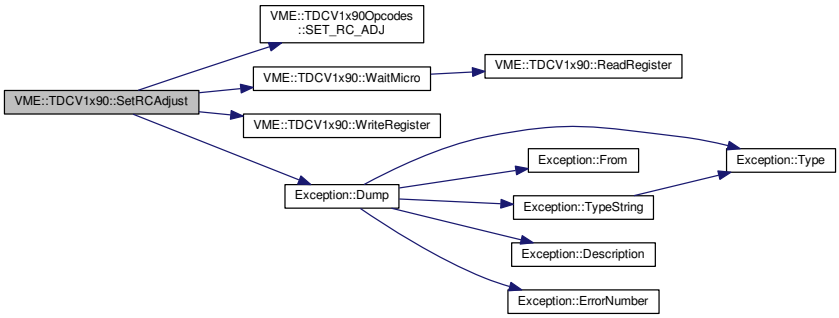
Here is the call graph for this function:



7.13.3.37 void VME::TDCV1x90::SetPol (uint16_t *word*)

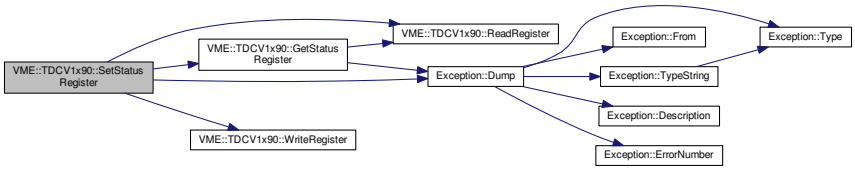
7.13.3.38 void VME::TDCV1x90::SetRCAdjust (int *tdc,* uint16_t *value*)

Here is the call graph for this function:



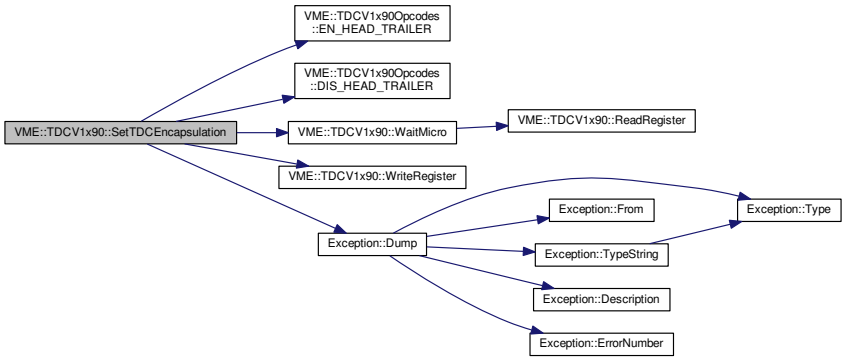
7.13.3.39 void VME::TDCV1x90::SetStatusRegister (stat_reg *reg,* bool *value*)

Here is the call graph for this function:



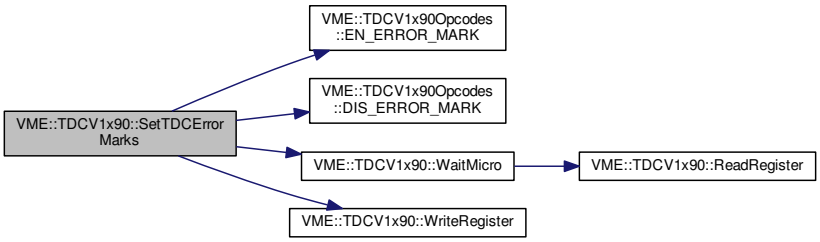
7.13.3.40 void VME::TDCV1x90::SetTDCEncapsulation (bool *mode*)

Here is the call graph for this function:



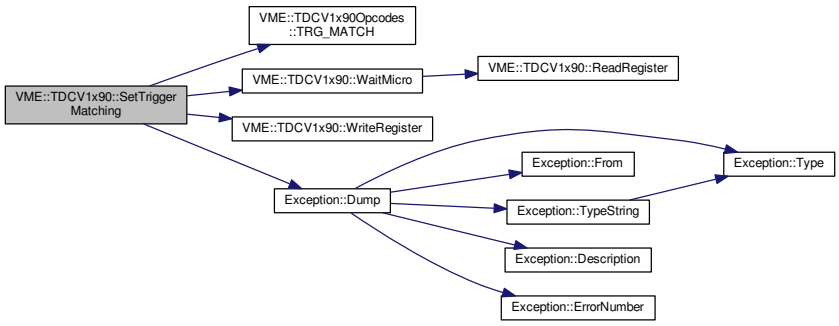
7.13.3.41 void VME::TDCV1x90::SetTDCErrorMarks (bool *mode*)

Here is the call graph for this function:



7.13.3.42 bool VME::TDCV1x90::SetTriggerMatching ( )

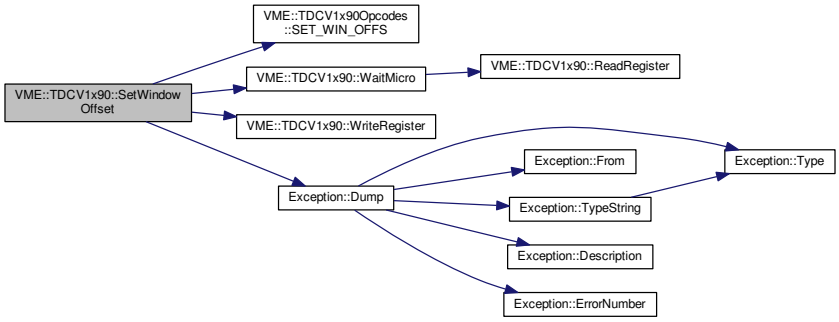
Here is the call graph for this function:



7.13.3.43 void VME::TDCV1x90::SetVerboseLevel (unsigned short *verb* = `1`) `[inline]`

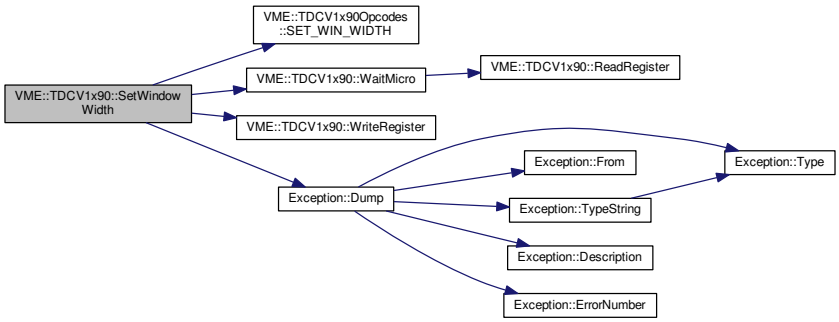
7.13.3.44 void VME::TDCV1x90::SetWindowOffset (int16_t *offs*)

Here is the call graph for this function:



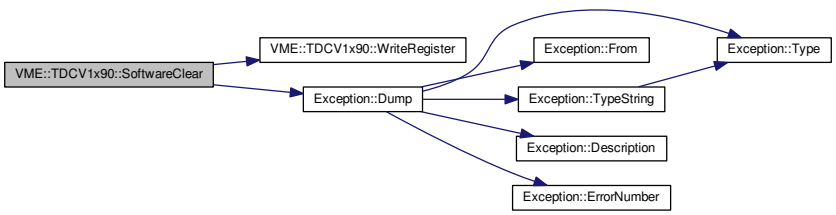
7.13.3.45 void VME::TDCV1x90::SetWindowWidth (uint16_t *width*)

Here is the call graph for this function:



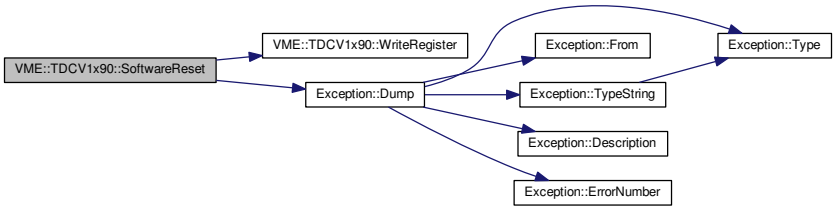
7.13.3.46 bool VME::TDCV1x90::SoftwareClear ( )

Here is the call graph for this function:



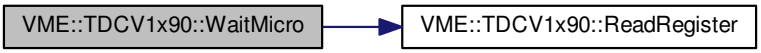
7.13.3.47 bool VME::TDCV1x90::SoftwareReset ( )

Here is the call graph for this function:



7.13.3.48 bool VME::TDCV1x90::WaitMicro (micro_handshake *mode*)

Here is the call graph for this function:



7.13.3.49 void VME::TDCV1x90::WriteRegister (mod_reg *addr,* uint16_t * *data*)

Write on register.

Write a 16-bit word in the register

Parameters

in	*addr*	register
in	*data*	word

7.13.3.50 void VME::TDCV1x90::WriteRegister (mod_reg *addr,* uint32_t * *data*)

Write on register.

Write a 32-bit word in the register

Parameters

in	*addr*	register
in	*data*	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 ettt

7.14.1 Field Documentation

7.14.1.1 uint32_t VME::trailead_t::ettt

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

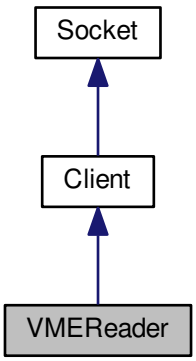
The documentation for this struct was generated from the following file:

- include/VME_TDCV1x90.h

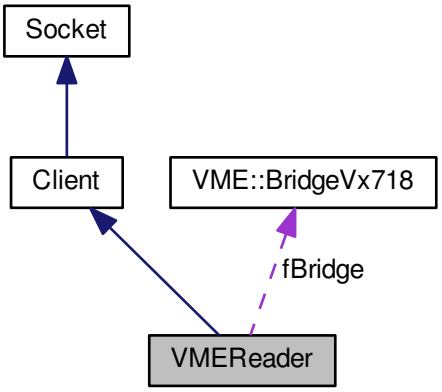
7.15 VMEReader Class Reference

`#include <VMEReader.h>`

Inheritance diagram for VMEReader:



Collaboration diagram for VMEReader:



Public Member Functions

- VMEReader (const char *device, VME::BridgeType type, bool on_socket=true)
- virtual ~VMEReader ()
- void AddTDC (uint32_t address)

  Add a TDC to handle.
- VME::TDCV1x90 * GetTDC (uint32_t address)

  Get a TDC on the VME bus Return a pointer to the TDC object, given its physical address on the VME bus.
- unsigned int GetRunNumber ()

  Ask the socket master a run number.
- void Abort ()

  Abort data collection for all modules on the bus handled by the bridge.

Private Types

- typedef std::map< uint32_t, VME::TDCV1x90 * > TDCCollection

  Mapper from physical VME addresses to pointers to TDC objects.

Private Attributes

- VME::BridgeVx718 * fBridge

  The VME bridge object to handle.
- TDCCollection fTDCCollection

  A set of pointers to TDC objects indexed by their physical VME address.
- bool fOnSocket

  Are we dealing with socket message passing?

Additional Inherited Members

7.15.1 Detailed Description

VME reader object to fetch events on a HPTDC board

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

4 May 2015

7.15.2 Member Typedef Documentation

7.15.2.1 typedef std::map<uint32_t,VME::TDCV1x90*> VMEReader::TDCCollection [private]

Mapper from physical VME addresses to pointers to TDC objects.

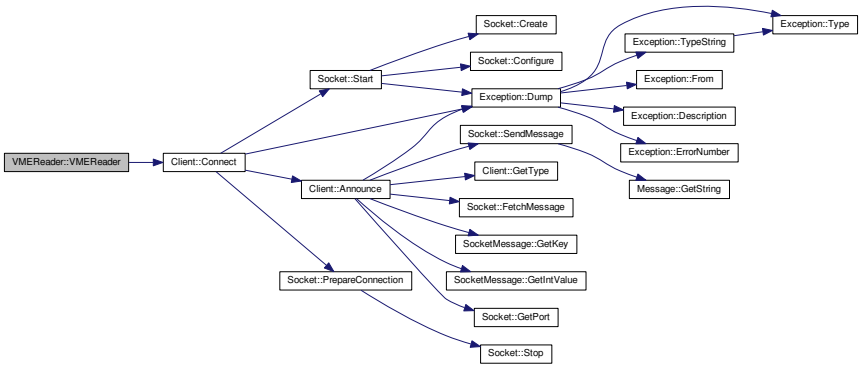
7.15.3 Constructor & Destructor Documentation

7.15.3.1 VMEReader::VMEReader (const char * *device*, VME::BridgeType *type*, bool *on_socket* = true)

Parameters

in	*device*	Path to the device (/dev/xxx)
in	*type*	Bridge model
in	*on_socket*	Are we trying to connect through the socket?

Here is the call graph for this function:



7.15.3.2 VMEReader::~VMEReader ( ) [virtual]

7.15.4 Member Function Documentation

7.15.4.1 void VMEReader::Abort ( )

Abort data collection for all modules on the bus handled by the bridge.

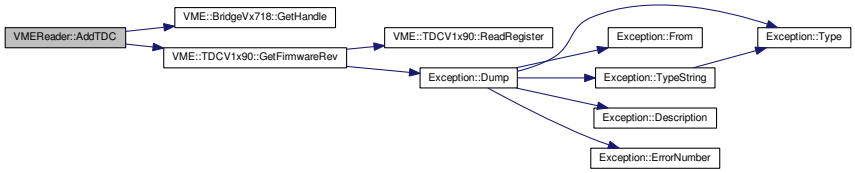
7.15.4.2 void VMEReader::AddTDC (uint32_t *address*)

Add a TDC to handle.

Parameters

in	*address*	32-bit address of the TDC module on the VME bus Create a new TDC handler for the VME bus

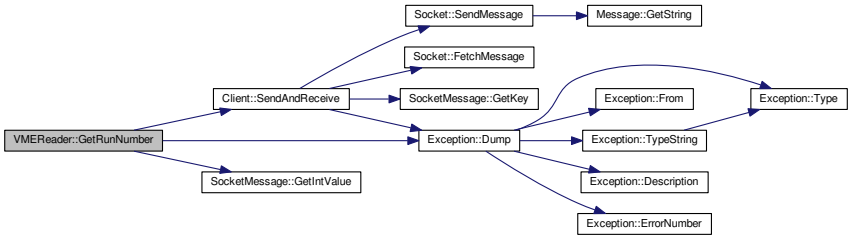
Here is the call graph for this function:



7.15.4.3 unsigned int VMEReader::GetRunNumber ( )

Ask the socket master a run number.

Here is the call graph for this function:



7.15.4.4 VME::TDCV1x90* VMEReader::GetTDC (uint32_t *address*) [inline]

Get a TDC on the VME bus Return a pointer to the TDC object, given its physical address on the VME bus.

7.15.5 Field Documentation

7.15.5.1 VME::BridgeVx718* VMEReader::fBridge [private]

The VME bridge object to handle.

7.15.5.2 bool VMEReader::fOnSocket [private]

Are we dealing with socket message passing?

7.15.5.3 TDCCollection VMEReader::fTDCCollection [private]

A set of pointers to TDC objects indexed by their physical VME address.

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

- include/VMEReader.h
- src/VMEReader.cpp

Index