2015 Test beam Run Control

Generated by Doxygen 1.8.9.1

Thu Apr 23 2015 16:05:25

Contents

1 Module Index **1**

1.1 Modules . . . 1

2 Hierarchical Index **3**

2.1 Class Hierarchy . . . 3

3 Data Structure Index **5**

3.1 Data Structures . . . 5

4 Module Documentation **7**

4.1 Socket communication objects . . . 7

4.1.1 Detailed Description . . . 7

4.1.2 Enumeration Type Documentation . . . 7

4.1.2.1 SocketType . . . 7

4.2 HPTDC chip control . . . 9

4.2.1 Detailed Description . . . 9

4.2.2 Enumeration Type Documentation . . . 9

4.2.2.1 DeadTime . . . 9

4.2.2.2 EdgeResolution . . . 10

4.2.2.3 EnabledError . . . 10

4.2.2.4 EventType . . . 10

4.2.2.5 WidthResolution . . . 11

5 Data Structure Documentation **13**

5.1 Client Class Reference . . . 13

5.1.1 Detailed Description . . . 14

5.1.2 Constructor & Destructor Documentation . . . 15

5.1.2.1 Client . . . 15

5.1.2.2 Client . . . 15

5.1.2.3 ∼Client . . . 15

5.1.3 Member Function Documentation . . . 15

5.1.3.1 Connect . . . 15

5.1.3.2 Disconnect . . . 15

5.1.3.3 GetType . . . 15
5.1.3.4 ParseMessage . . . 15
5.1.3.5 Receive . . . 15
5.1.3.6 Send . . . 15
5.2 Exception Class Reference . . . 16
5.2.1 Detailed Description . . . 16
5.2.2 Constructor & Destructor Documentation . . . 16
5.2.2.1 Exception . . . 16
5.2.2.2 Exception . . . 16
5.2.2.3 ∼Exception . . . 16
5.2.3 Member Function Documentation . . . 17
5.2.3.1 Description . . . 17
5.2.3.2 Dump . . . 17
5.2.3.3 ErrorNumber . . . 17
5.2.3.4 From . . . 18
5.2.3.5 Type . . . 18
5.2.3.6 TypeString . . . 18
5.3 file_header_t Struct Reference . . . 19
5.3.1 Detailed Description . . . 19
5.3.2 Field Documentation . . . 20
5.3.2.1 config . . . 20
5.3.2.2 magic . . . 20
5.3.2.3 run_id . . . 20
5.3.2.4 spill_id . . . 20
5.4 FPGAHandler Class Reference . . . 20
5.4.1 Detailed Description . . . 21
5.4.2 Constructor & Destructor Documentation . . . 22
5.4.2.1 FPGAHandler . . . 22
5.4.2.2 ∼FPGAHandler . . . 22
5.4.3 Member Function Documentation . . . 22
5.4.3.1 CloseFile . . . 22
5.4.3.2 GetConfiguration . . . 22
5.4.3.3 GetFilename . . . 22
5.4.3.4 GetType . . . 22
5.4.3.5 OpenFile . . . 22
5.4.3.6 ReadBuffer . . . 22
5.4.3.7 SetConfiguration . . . 22
5.5 HTTPMessage Class Reference . . . 22
5.5.1 Detailed Description . . . 23
5.5.2 Constructor & Destructor Documentation . . . 24

5.5.2.1 HTTPMessage . . . 24
5.5.2.2 HTTPMessage . . . 24
5.5.3 Member Function Documentation . . . 24
5.5.3.1 Decode . . . 25
5.5.3.2 Dump . . . 25
5.5.3.3 Encode . . . 25
5.5.3.4 GetKey . . . 25
5.6 ListenerInfo Struct Reference . . . 25
5.6.1 Detailed Description . . . 25
5.6.2 Field Documentation . . . 26
5.6.2.1 name . . . 26
5.6.2.2 type . . . 26
5.7 Message Class Reference . . . 26
5.7.1 Detailed Description . . . 27
5.7.2 Constructor & Destructor Documentation . . . 27
5.7.2.1 Message . . . 27
5.7.2.2 Message . . . 27
5.7.2.3 Message . . . 27
5.7.2.4 ∼Message . . . 27
5.7.3 Member Function Documentation . . . 27
5.7.3.1 Dump . . . 27
5.7.3.2 GetKey . . . 27
5.7.3.3 GetString . . . 27
5.7.3.4 IsFromWeb . . . 28
5.7.4 Field Documentation . . . 28
5.7.4.1 fString . . . 28
5.8 Messenger Class Reference . . . 28
5.8.1 Detailed Description . . . 29
5.8.2 Constructor & Destructor Documentation . . . 29
5.8.2.1 Messenger . . . 29
5.8.2.2 Messenger . . . 29
5.8.2.3 ∼Messenger . . . 29
5.8.3 Member Function Documentation . . . 29
5.8.3.1 Broadcast . . . 29
5.8.3.2 Connect . . . 30
5.8.3.3 Disconnect . . . 30
5.8.3.4 GetType . . . 30
5.8.3.5 Receive . . . 30
5.8.3.6 Send . . . 30
5.9 Socket Class Reference . . . 30

5.9.1 Detailed Description . . . 32
5.9.2 Member Typedef Documentation . . . 32
5.9.2.1 SocketCollection . . . 32
5.9.3 Constructor & Destructor Documentation . . . 32
5.9.3.1 Socket . . . 32
5.9.3.2 Socket . . . 32
5.9.3.3 ~Socket . . . 32
5.9.4 Member Function Documentation . . . 32
5.9.4.1 AcceptConnections . . . 32
5.9.4.2 Bind . . . 33
5.9.4.3 DumpConnected . . . 33
5.9.4.4 FetchMessage . . . 33
5.9.4.5 GetPort . . . 33
5.9.4.6 GetSocketId . . . 33
5.9.4.7 GetSocketType . . . 33
5.9.4.8 IsWebSocket . . . 34
5.9.4.9 Listen . . . 34
5.9.4.10 PrepareConnection . . . 34
5.9.4.11 SelectConnections . . . 34
5.9.4.12 SendMessage . . . 34
5.9.4.13 SetPort . . . 34
5.9.4.14 SetSocketId . . . 34
5.9.4.15 Start . . . 34
5.9.4.16 Stop . . . 35
5.9.5 Field Documentation . . . 35
5.9.5.1 fBuffer . . . 35
5.9.5.2 fMaster . . . 35
5.9.5.3 fPort . . . 35
5.9.5.4 fReadFds . . . 35
5.9.5.5 fSocketsConnected . . . 35
5.10 SocketMessage Class Reference . . . 35
5.10.1 Detailed Description . . . 37
5.10.2 Constructor & Destructor Documentation . . . 37
5.10.2.1 SocketMessage . . . 37
5.10.2.2 SocketMessage . . . 37
5.10.2.3 SocketMessage . . . 37
5.10.2.4 SocketMessage . . . 37
5.10.2.5 SocketMessage . . . 37
5.10.2.6 SocketMessage . . . 37
5.10.2.7 SocketMessage . . . 38

5.10.2.8 SocketMessage . . . 38
5.10.2.9 SocketMessage . . . 38
5.10.2.10 SocketMessage . . . 39
5.10.2.11 SocketMessage . . . 39
5.10.2.12 ∼SocketMessage . . . 39
5.10.3 Member Function Documentation . . . 39
5.10.3.1 Dump . . . 39
5.10.3.2 GetIntValue . . . 39
5.10.3.3 GetKey . . . 39
5.10.3.4 GetString . . . 40
5.10.3.5 GetValue . . . 40
5.10.3.6 GetVectorValue . . . 40
5.10.3.7 SetKeyValue . . . 41
5.10.3.8 SetKeyValue . . . 41
5.10.3.9 SetKeyValue . . . 41
5.10.3.10 SetKeyValue . . . 41
5.11 TDCConfiguration Class Reference . . . 42
5.11.1 Detailed Description . . . 44
5.11.2 Constructor & Destructor Documentation . . . 44
5.11.2.1 TDCConfiguration . . . 44
5.11.2.2 ∼TDCConfiguration . . . 44
5.11.3 Member Function Documentation . . . 44
5.11.3.1 Dump . . . 44
5.11.3.2 GetChannelOffset . . . 44
5.11.3.3 GetCoarseCountOffset . . . 44
5.11.3.4 GetDeadTime . . . 45
5.11.3.5 GetDLLAdjustment . . . 45
5.11.3.6 GetEdgeResolution . . . 45
5.11.3.7 GetEdgesPairing . . . 45
5.11.3.8 GetEnableError . . . 45
5.11.3.9 GetEnableErrorBypass . . . 45
5.11.3.10 GetEnableErrorMark . . . 45
5.11.3.11 GetEnableJTAGReadout . . . 45
5.11.3.12 GetEnableReadoutOccupancy . . . 45
5.11.3.13 GetEnableReadoutSeparator . . . 45
5.11.3.14 GetEnableSerial . . . 45
5.11.3.15 GetLeadingMode . . . 45
5.11.3.16 GetMaxEventSize . . . 45
5.11.3.17 GetNumWords . . . 46
5.11.3.18 GetRCAdjustment . . . 46

5.11.3.19 GetRejectFIFOFull . . . 46
5.11.3.20 GetSetupParity . . . 46
5.11.3.21 GetTrailingMode . . . 46
5.11.3.22 GetTriggerCountOffset . . . 46
5.11.3.23 GetTriggerLatency . . . 46
5.11.3.24 GetTriggerMatchingMode . . . 47
5.11.3.25 GetVernierOffset . . . 47
5.11.3.26 GetWidthResolution . . . 47
5.11.3.27 GetWord . . . 47
5.11.3.28 SetAllChannelsOffset . . . 47
5.11.3.29 SetAllTapsDLLAdjustment . . . 47
5.11.3.30 SetChannelOffset . . . 48
5.11.3.31 SetCoarseCountOffset . . . 48
5.11.3.32 SetConstantValues . . . 48
5.11.3.33 SetDeadTime . . . 48
5.11.3.34 SetDLLAdjustment . . . 48
5.11.3.35 SetEdgeResolution . . . 48
5.11.3.36 SetEdgesPairing . . . 48
5.11.3.37 SetEnableError . . . 48
5.11.3.38 SetEnableErrorBypass . . . 48
5.11.3.39 SetEnableErrorMark . . . 49
5.11.3.40 SetEnableJTAGReadout . . . 49
5.11.3.41 SetEnableReadoutOccupancy . . . 49
5.11.3.42 SetEnableReadoutSeparator . . . 49
5.11.3.43 SetEnableSerial . . . 49
5.11.3.44 SetLeadingMode . . . 49
5.11.3.45 SetMaxEventSize . . . 50
5.11.3.46 SetRCAdjustment . . . 50
5.11.3.47 SetRejectFIFOFull . . . 50
5.11.3.48 SetSetupParity . . . 50
5.11.3.49 SetTrailingMode . . . 50
5.11.3.50 SetTriggerCountOffset . . . 50
5.11.3.51 SetTriggerMatchingMode . . . 50
5.11.3.52 SetVernierOffset . . . 50
5.11.3.53 SetWidthResolution . . . 50
5.11.3.54 SetWord . . . 50
5.12 TDCEvent Class Reference . . . 50
5.12.1 Detailed Description . . . 51
5.12.2 Constructor & Destructor Documentation . . . 51
5.12.2.1 TDCEvent . . . 51

5.12.2.2 ~TDCEvent . . . 51
5.12.3 Member Function Documentation . . . 51
5.12.3.1 GetBunchId . . . 52
5.12.3.2 GetErrorFlags . . . 52
5.12.3.3 GetEventId . . . 52
5.12.3.4 GetLeadingTime . . . 52
5.12.3.5 GetTDCId . . . 53
5.12.3.6 GetTrailingTime . . . 53
5.12.3.7 GetType . . . 53
5.12.3.8 GetWidth . . . 54
5.12.3.9 GetWordCount . . . 54
5.13 USBHandler Class Reference . . . 55
5.13.1 Detailed Description . . . 56
5.13.2 Constructor & Destructor Documentation . . . 56
5.13.2.1 USBHandler . . . 56
5.13.2.2 ~USBHandler . . . 56
5.13.3 Member Function Documentation . . . 56
5.13.3.1 DumpDevice . . . 56
5.13.3.2 Fetch . . . 56
5.13.3.3 Init . . . 56
5.13.3.4 Write . . . 56

Index **57**

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Socket communication objects 7
HPTDC chip control 9

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

- Exception . . . 16
- file_header_t . . . 19
- ListenerInfo . . . 25
- Message . . . 26
  - HTTPMessage . . . 22
  - SocketMessage . . . 35
- Socket . . . 30
  - Client . . . 13
    - FPGAHandler . . . 20
  - Messenger . . . 28
- TDCConfiguration . . . 42
- TDCEvent . . . 50
- USBHandler . . . 55
  - FPGAHandler . . . 20

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Client
: Base client object for the socket . . . 13

Exception
: A simple exception handler . . . 16

file_header_t
: Header to the output files . . . 19

FPGAHandler
: Driver for timing detectors' FPGA readout . . . 20

HTTPMessage
: Message to be transmitted through a WebSocket protocol . . . 22

ListenerInfo
: Information on a socket's listener . . . 25

Message
: Base socket message type . . . 26

Messenger
: Base master object for the socket . . . 28

Socket
: Base socket object from which clients/master from a socket inherit . . . 30

SocketMessage
: Socket-passed message type . . . 35

TDCConfiguration
: Setup word to be sent to the HPTDC chip . . . 42

TDCEvent
: HPTDC event parser . . . 50

USBHandler
: Generic USB communication handler . . . 55

Chapter 4

Module Documentation

4.1 Socket communication objects

Data Structures

- class Client

  Base client object for the socket.
- class HTTPMessage

  Message to be transmitted through a WebSocket protocol.
- struct ListenerInfo

  Information on a socket's listener.
- 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.

4.1.1 Detailed Description

4.1.2 Enumeration Type Documentation

4.1.2.1 enum Socket::SocketType

Type of actor playing a role on the socket.

Enumerator

INVALID

MASTER

WEBSOCKET_CLIENT

CLIENT

DETECTOR

4.2 HPTDC chip control

Data Structures

- class TDCConfiguration

  Setup word to be sent to the HPTDC chip.
- class TDCEvent

  HPTDC event parser.

Enumerations

- enum TDCConfiguration::EdgeResolution { TDCConfiguration::E_100ps =0, TDCConfiguration::E_200ps, TDCConfiguration::E_400ps, TDCConfiguration::E_800ps, TDCConfiguration::E_1p6ns, TDCConfiguration::E_3p12ns, TDCConfiguration::E_6p25ns, TDCConfiguration::E_12p5ns }
- enum TDCConfiguration::DeadTime { TDCConfiguration::DT_5ns =0, TDCConfiguration::DT_10ns, TDCConfiguration::DT_30ns, TDCConfiguration::DT_100ns }
- enum TDCConfiguration::WidthResolution { TDCConfiguration::W_100ps =0, TDCConfiguration::W_200ps, TDCConfiguration::W_400ps, TDCConfiguration::W_800ps, TDCConfiguration::W_1p6ns, TDCConfiguration::W_3p2ns, TDCConfiguration::W_6p25ns, TDCConfiguration::W_12p5ns, TDCConfiguration::W_25ns, TDCConfiguration::W_50ns, TDCConfiguration::W_100ns, TDCConfiguration::W_200ns, TDCConfiguration::W_400ns, TDCConfiguration::W_800ns }
- enum TDCConfiguration::EnabledError { TDCConfiguration::VernierError =0x1, TDCConfiguration::CoarseError =0x2, TDCConfiguration::ChannelSelectError =0x4, TDCConfiguration::L1BufferParityError =0x8, TDCConfiguration::TriggerFIFOParityError =0x10, TDCConfiguration::TriggerMatchingError =0x20, TDCConfiguration::ReadoutFIFOParityError =0x40, TDCConfiguration::ReadoutStateError =0x80, TDCConfiguration::SetupParityError =0x100, TDCConfiguration::ControlParityError =0x200, TDCConfiguration::JTAGInstructionParityError =0x400 }
- enum TDCEvent::EventType { TDCEvent::Invalid =-1, TDCEvent::GroupHeader =0, TDCEvent::GroupTrailer, TDCEvent::TDCHeader, TDCEvent::TDCTrailer, TDCEvent::LeadingEdge, TDCEvent::TrailingEdge, TDCEvent::Error, TDCEvent::Debug }

4.2.1 Detailed Description

4.2.2 Enumeration Type Documentation

4.2.2.1 enum TDCConfiguration::DeadTime

Enumerator

DT_5ns

DT_10ns

DT_30ns

DT_100ns

4.2.2.2 enum TDCConfiguration::EdgeResolution

Enumerator

E_100ps

E_200ps

E_400ps

E_800ps

E_1p6ns

E_3p12ns

E_6p25ns

E_12p5ns

4.2.2.3 enum TDCConfiguration::EnabledError

Enumerator

VernierError

CoarseError

ChannelSelectError

L1BufferParityError

TriggerFIFOParityError

TriggerMatchingError

ReadoutFIFOParityError

ReadoutStateError

SetupParityError

ControlParityError

JTAGInstructionParityError

4.2.2.4 enum TDCEvent::EventType

Enumerator

Invalid

GroupHeader

GroupTrailer

TDCHeader

TDCTrailer

LeadingEdge

TrailingEdge

Error

Debug

4.2.2.5 enum TDCConfiguration::WidthResolution

Enumerator

W_100ps

W_200ps

W_400ps

W_800ps

W_1p6ns

W_3p2ns

W_6p25ns

W_12p5ns

W_25ns

W_50ns

W_100ns

W_200ns

W_400ns

W_800ns

Chapter 5

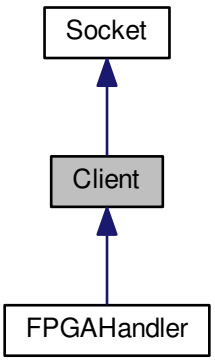
Data Structure Documentation

5.1 Client Class Reference

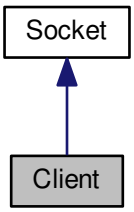
Base client object for the socket.

`#include <Client.h>`

Inheritance diagram for Client:



Collaboration diagram for Client:



Public Member Functions

- Client ()

  General void client constructor.
- Client (int port)

  Bind a socket client to a given port.
- virtual ~Client ()
- bool Connect ()

  Bind this client to the socket.
- void Disconnect ()

  Unbind this client from the socket.
- void Send (const Message &m) const

  Send a message to the master through the socket.
- void Receive ()

  Receive a socket message from the master.
- virtual void ParseMessage (const SocketMessage &m)

  Parse a SocketMessage received from the master.
- virtual SocketType GetType () const

  Socket actor type retrieval method.

Additional Inherited Members

5.1.1 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

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Client::Client ( ) [inline]

General void client constructor.

5.1.2.2 Client::Client (int *port*)

Bind a socket client to a given port.

5.1.2.3 virtual Client::~Client ( ) [virtual]

5.1.3 Member Function Documentation

5.1.3.1 bool Client::Connect ( )

Bind this client to the socket.

5.1.3.2 void Client::Disconnect ( )

Unbind this client from the socket.

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

Socket actor type retrieval method.

Reimplemented in FPGAHandler.

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

Parse a SocketMessage received from the master.

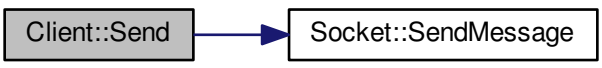
5.1.3.5 void Client::Receive ( )

Receive a socket message from the master.

5.1.3.6 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:



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

- include/Client.h

5.2 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

5.2.1 Detailed Description

A simple exception handler.

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

24 Mar 2015

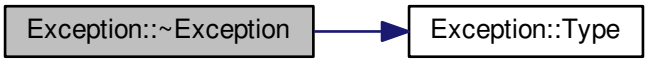
5.2.2 Constructor & Destructor Documentation

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

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

5.2.2.3 Exception::~Exception ( ) [inline]

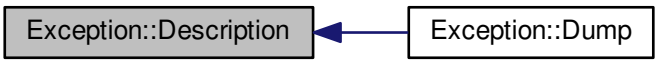
Here is the call graph for this function:



5.2.3 Member Function Documentation

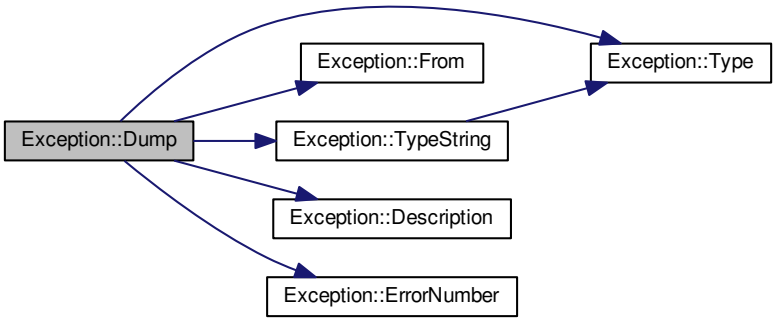
5.2.3.1 std::string Exception::Description ( ) const [inline]

Here is the caller graph for this function:



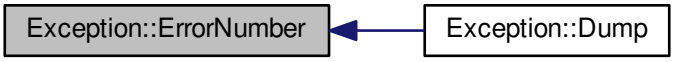
5.2.3.2 void Exception::Dump (std::ostream & *os* = std::cerr) const [inline]

Here is the call graph for this function:



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

Here is the caller graph for this function:

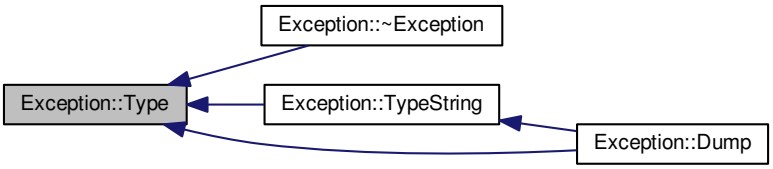


5.2.3.4 std::string Exception::From ( ) const [inline]

Here is the caller graph for this function:

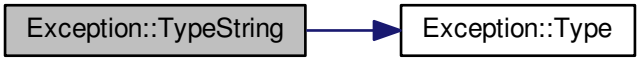
5.2.3.5 ExceptionType Exception::Type ( ) const [inline]

Here is the caller graph for this function:

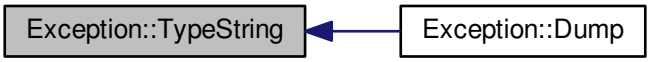


5.2.3.6 std::string Exception::TypeString ( ) const [inline]

Here is the call graph for this function:



Here is the caller graph for this function:



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

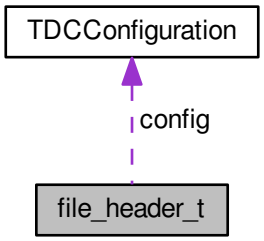
- include/Exception.h

5.3 file_header_t Struct Reference

Header to the output files.

`#include <FPGAHandler.h>`

Collaboration diagram for file_header_t:



Data Fields

- uint32_t magic
- uint32_t run_id
- uint32_t spill_id
- TDCConfiguration config

5.3.1 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

5.3.2 Field Documentation

5.3.2.1 TDCConfiguration file_header_t::config

5.3.2.2 uint32_t file_header_t::magic

5.3.2.3 uint32_t file_header_t::run_id

5.3.2.4 uint32_t file_header_t::spill_id

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

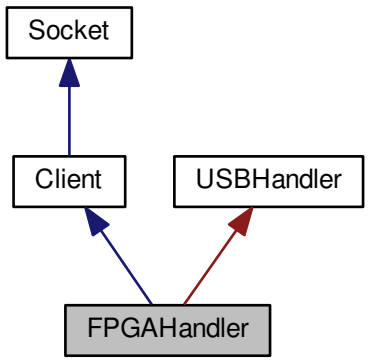
- include/FPGAHandler.h

5.4 FPGAHandler Class Reference

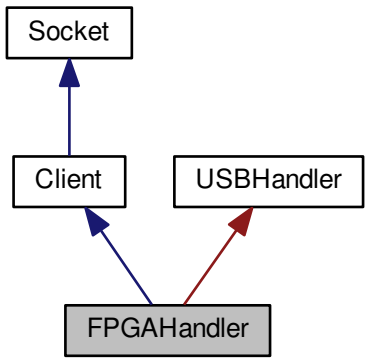
Driver for timing detectors' FPGA readout.

`#include <FPGAHandler.h>`

Inheritance diagram for FPGAHandler:



Collaboration diagram for FPGAHandler:



Public Member Functions

- FPGAHandler (int port, const char *dev)

  Bind to a FPGA through the USB protocol, and to the socket.
- virtual ~FPGAHandler ()
- void OpenFile ()

  Open an output file to store header/HPTDC events.
- void CloseFile ()

  Close a previously opened output file used to store header/HPTDC events.
- std::string GetFilename () const

  Retrieve the file name used to store data collected from the FPGA.
- void SetConfiguration (const TDCConfiguration &c)

  Submit the HPTDC setup word as a TDCConfiguration object.
- TDCConfiguration GetConfiguration ()

  Retrieve the HPTDC setup word as a TDCConfiguration object.
- void ReadBuffer ()
- SocketType GetType () const

  Socket actor type retrieval method.

Additional Inherited Members

5.4.1 Detailed Description

Driver for timing detectors' FPGA readout.

Main driver for a homebrew FPGA designed for the timing detectors' HPTDC chip readout.

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

14 Apr 2015

5.4.2 Constructor & Destructor Documentation

5.4.2.1 FPGAHandler::FPGAHandler (int *port,* const char * *dev*)

Bind to a FPGA through the USB protocol, and to the socket.

5.4.2.2 virtual FPGAHandler::~FPGAHandler ( ) [virtual]

5.4.3 Member Function Documentation

5.4.3.1 void FPGAHandler::CloseFile ( )

Close a previously opened output file used to store header/HPTDC events.

5.4.3.2 TDCConfiguration FPGAHandler::GetConfiguration ( ) [inline]

Retrieve the HPTDC setup word as a TDCConfiguration object.

5.4.3.3 std::string FPGAHandler::GetFilename ( ) const [inline]

Retrieve the file name used to store data collected from the FPGA.

5.4.3.4 SocketType FPGAHandler::GetType ( ) const [inline], [virtual]

Socket actor type retrieval method.

Reimplemented from Client.

5.4.3.5 void FPGAHandler::OpenFile ( )

Open an output file to store header/HPTDC events.

5.4.3.6 void FPGAHandler::ReadBuffer ( )

5.4.3.7 void FPGAHandler::SetConfiguration (const TDCConfiguration & *c*) [inline]

Submit the HPTDC setup word as a TDCConfiguration object.

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

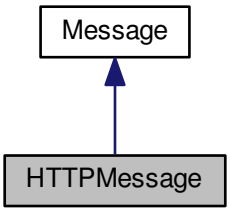
- include/FPGAHandler.h

5.5 HTTPMessage Class Reference

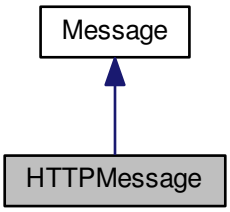
Message to be transmitted through a WebSocket protocol.

`#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

Additional Inherited Members

5.5.1 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

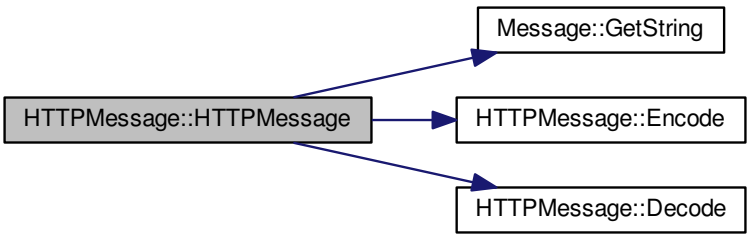
Date

1 Apr 2015

5.5.2 Constructor & Destructor Documentation

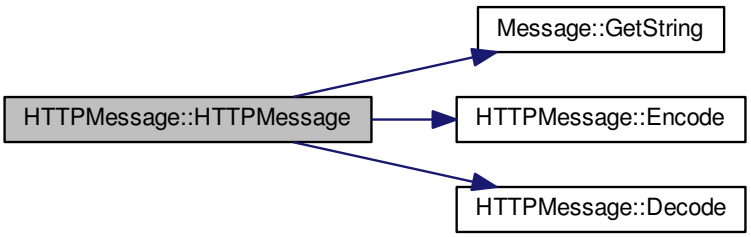
5.5.2.1 HTTPMessage::HTTPMessage (WebSocket * *ws*, Message *m*, MessageAction *a*) [inline]

Here is the call graph for this function:



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

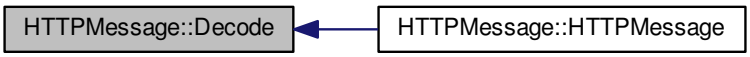
Here is the call graph for this function:



5.5.3 Member Function Documentation

5.5.3.1 void HTTPMessage::Decode ( ) [inline]

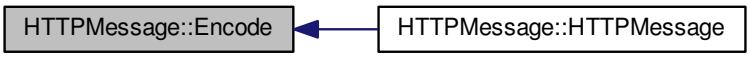
Here is the caller graph for this function:



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

5.5.3.3 void HTTPMessage::Encode ( ) [inline]

Here is the caller graph for this function:



5.5.3.4 MessageKey HTTPMessage::GetKey ( ) const [inline]

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

- include/HTTPMessage.h

5.6 ListenerInfo Struct Reference

Information on a socket's listener.

`#include <Messenger.h>`

Data Fields

- std::string name
- Socket::SocketType type

5.6.1 Detailed Description

Information on a socket's listener.

Structure handling its name and type for any listener/client to be used in the socket management parts of this code.

5.6.2 Field Documentation

5.6.2.1 std::string ListenerInfo::name

5.6.2.2 Socket::SocketType ListenerInfo::type

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

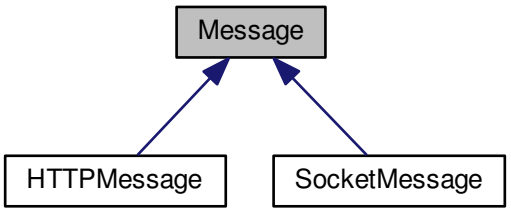
- include/Messenger.h

5.7 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

5.7.1 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

5.7.2 Constructor & Destructor Documentation

5.7.2.1 Message::Message ( ) [inline]

Void message constructor.

5.7.2.2 Message::Message (const char * *msg*) [inline]

Construct a message from a string.

5.7.2.3 Message::Message (std::string *msg*) [inline]

Construct a message from a string.

5.7.2.4 virtual Message::~Message ( ) [inline], [virtual]

5.7.3 Member Function Documentation

5.7.3.1 void Message::Dump (std::ostream & *os* = std::cout) const [inline]

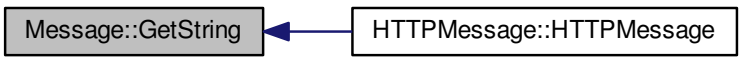
5.7.3.2 MessageKey Message::GetKey ( ) const [inline]

Placeholder for the MessageKey retrieval method.

5.7.3.3 std::string Message::GetString ( ) const [inline]

Retrieve the string carried by this message as a whole.

Here is the caller graph for this function:



5.7.3.4 bool Message::IsFromWeb ( ) const [inline]

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

5.7.4 Field Documentation

5.7.4.1 std::string Message::fString [protected]

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

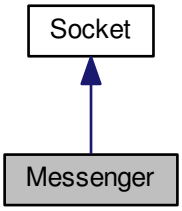
- include/Message.h

5.8 Messenger Class Reference

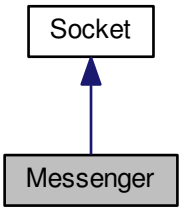
Base master object for the socket.

`#include <Messenger.h>`

Inheritance diagram for Messenger:



Collaboration diagram for Messenger:



Public Member Functions

- Messenger ()

Build a void master object or socket actor.

- Messenger (int port)

  Build a master object to control the socket.
- ~Messenger ()
- bool Connect ()

  Connect the master to the socket.
- void Disconnect ()

  Remove the master and destroy the socket.
- void Send (const Message &m, int sid) const

  Send any type of message to any client.
- void Receive ()

  Handle a message reception from a client.
- void Broadcast (const Message &m) const

  Emit a message to all clients connected through the socket.
- SocketType GetType () const

  Socket actor type retrieval method.

Additional Inherited Members

5.8.1 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

5.8.2 Constructor & Destructor Documentation

5.8.2.1 Messenger::Messenger ( )

Build a void master object or socket actor.

5.8.2.2 Messenger::Messenger (int *port*)

Build a master object to control the socket.

5.8.2.3 Messenger::~Messenger ( )

5.8.3 Member Function Documentation

5.8.3.1 void Messenger::Broadcast (const Message & *m*) const

Emit a message to all clients connected through the socket.

Parameters

in	*m*	Message to transmit

5.8.3.2 bool Messenger::Connect ( )

Connect the master to the socket.

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

5.8.3.3 void Messenger::Disconnect ( )

Remove the master and destroy the socket.

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

5.8.3.4 SocketType Messenger::GetType ( ) const [inline]

Socket actor type retrieval method.

5.8.3.5 void Messenger::Receive ( )

Handle a message reception from a client.

5.8.3.6 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

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

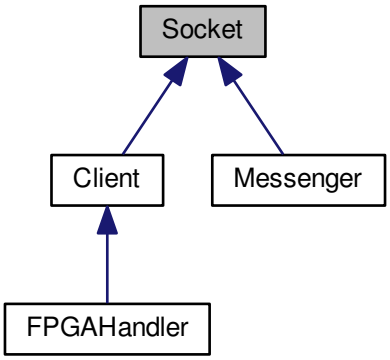
- include/Messenger.h

5.9 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()

5.9.1 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

5.9.2 Member Typedef Documentation

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

5.9.3 Constructor & Destructor Documentation

5.9.3.1 Socket::Socket ( ) [inline]

5.9.3.2 Socket::Socket (int *port*)

5.9.3.3 virtual Socket::~Socket ( ) [virtual]

5.9.4 Member Function Documentation

5.9.4.1 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

5.9.4.2 void Socket::Bind ( ) [protected]

Bind a name to a socket.

Returns

Success of the operation

5.9.4.3 void Socket::DumpConnected ( ) const

5.9.4.4 Message Socket::FetchMessage (int *id* = -1) const [protected]

Receive a message from a socket.

Returns

Received message as a std::string

5.9.4.5 int Socket::GetPort ( ) const [inline]

Retrieve the port used for this socket.

5.9.4.6 int Socket::GetSocketId ( ) const [inline]

5.9.4.7 SocketType Socket::GetSocketType (int *sid*) const [inline]

Here is the caller graph for this function:

5.9.4.8 bool Socket::IsWebSocket (int *sid*) const [inline]

Here is the call graph for this function:

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

Listen to incoming messages.

Set the socket to listen to any message coming from outside

5.9.4.10 void Socket::PrepareConnection ( ) [protected]

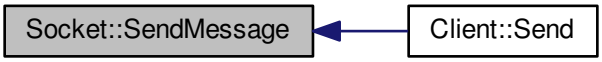
5.9.4.11 void Socket::SelectConnections ( )

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

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

Send a message on a socket.

Here is the caller graph for this function:



5.9.4.13 void Socket::SetPort (int *port*) [inline]

5.9.4.14 void Socket::SetSocketId (int *sid*) [inline]

5.9.4.15 bool Socket::Start ( ) [protected]

Start the socket.

Launch all mandatory operations to set the socket to be used

Returns

Success of the operation

5.9.4.16 void Socket::Stop ( )

Terminates the socket and all attached communications.

5.9.5 Field Documentation

5.9.5.1 char Socket::fBuffer[MAX_WORD_LENGTH] [protected]

5.9.5.2 fd_set Socket::fMaster [protected]

Master file descriptor list.

5.9.5.3 int Socket::fPort [protected]

5.9.5.4 fd_set Socket::fReadFds [protected]

Temp file descriptor list for select()

5.9.5.5 SocketCollection Socket::fSocketsConnected [protected]

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

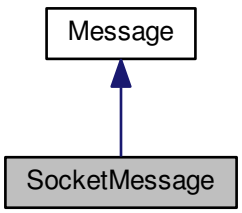
- include/Socket.h

5.10 SocketMessage Class Reference

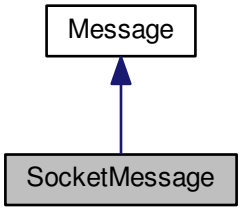
Socket-passed message type.

`#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

Additional Inherited Members

5.10.1 Detailed Description

Socket-passed message type.

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

26 Mar 2015

5.10.2 Constructor & Destructor Documentation

5.10.2.1 SocketMessage::SocketMessage ( ) [inline]

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

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

5.10.2.4 SocketMessage::SocketMessage (std::string *msg_s*) [inline]

5.10.2.5 SocketMessage::SocketMessage (const MessageKey & *key*) [inline]

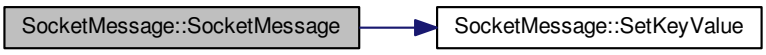
Construct a socket message out of a key.

Here is the call graph for this function:

5.10.2.6 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:



5.10.2.7 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:

5.10.2.8 SocketMessage::SocketMessage (const MessageKey & *key*, const int *value*) [inline]

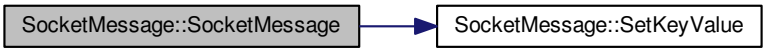
Construct a socket message out of a key and an integer-type value.

Here is the call graph for this function:

5.10.2.9 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:



5.10.2.10 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:

5.10.2.11 SocketMessage::SocketMessage (MessageMap *msg_m*) [inline]

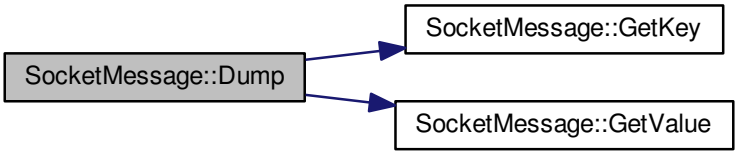
Construct a socket message out of a map of key/string-type value.

5.10.2.12 SocketMessage::~SocketMessage ( ) [inline]

5.10.3 Member Function Documentation

5.10.3.1 void SocketMessage::Dump (std::ostream & *os* = std::cout) const [inline]

Here is the call graph for this function:



5.10.3.2 int SocketMessage::GetIntValue ( ) const [inline]

Extract the message's integer value.

5.10.3.3 MessageKey SocketMessage::GetKey ( ) const [inline]

Extract the message's key.

Here is the caller graph for this function:

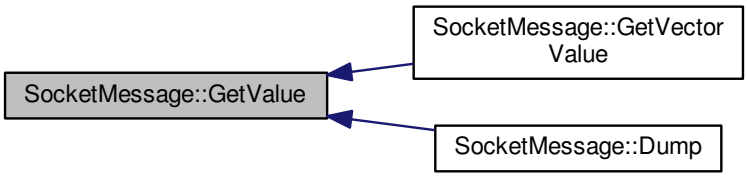
5.10.3.4 std::string SocketMessage::GetString ( ) const [inline]

Extract the whole key:value message.

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

Extract the message's string value.

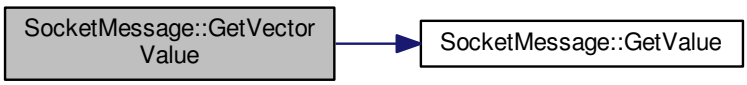
Here is the caller graph for this function:



5.10.3.6 VectorValue SocketMessage::GetVectorValue ( ) const [inline]

Extract the message's vector of string value.

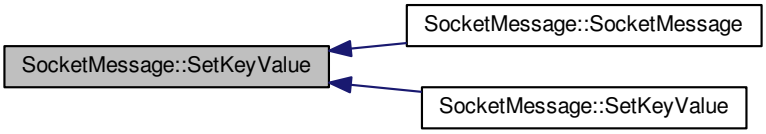
Here is the call graph for this function:



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

String-valued message.

Here is the caller graph for this function:



5.10.3.8 void SocketMessage::SetKeyValue (const MessageKey & *key*, int *int_value*) [inline]

Send an integer-valued message.

Here is the call graph for this function:

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

Float-valued message.

Here is the call graph for this function:

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

Double-valued message.

Here is the call graph for this function:

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

- include/SocketMessage.h

5.11 TDCConfiguration Class Reference

Setup word to be sent to the HPTDC chip.

`#include <TDCConfiguration.h>`

Public Types

- enum EdgeResolution {
  E_100ps =0, E_200ps, E_400ps, E_800ps,
  E_1p6ns, E_3p12ns, E_6p25ns, E_12p5ns }
- enum DeadTime { DT_5ns =0, DT_10ns, DT_30ns, DT_100ns }
- enum WidthResolution {
  W_100ps =0, W_200ps, W_400ps, W_800ps,
  W_1p6ns, W_3p2ns, W_6p25ns, W_12p5ns,
  W_25ns, W_50ns, W_100ns, W_200ns,
  W_400ns, W_800ns }
- enum EnabledError {
  VernierError =0x1, CoarseError =0x2, ChannelSelectError =0x4, L1BufferParityError =0x8,
  TriggerFIFOParityError =0x10, TriggerMatchingError =0x20, ReadoutFIFOParityError =0x40, ReadoutState↩
  Error =0x80,
  SetupParityError =0x100, ControlParityError =0x200, JTAGInstructionParityError =0x400 }

Public Member Functions

- TDCConfiguration ()
- virtual ~TDCConfiguration ()
- void SetWord (const unsigned int i, const word_t word)

  Set one bit(s) subset in the setup word.
- word_t GetWord (const unsigned int i) const

  Retrieve one subset from the setup word.
- uint8_t GetNumWords () const

  Number of words in the configuration.
- void SetEnableErrorMark (bool em)

  Mark events with error if global error signal is set.
- bool GetEnableErrorMark () const
- void SetEnableErrorBypass (bool eb)

  Bypass TDC chip if global error signal is set.

- bool GetEnableErrorBypass () const
- void SetEnableError (const uint16_t &err)

  Enable internal error types for generation of global error signals.
- uint16_t GetEnableError () const
- void SetEnableSerial (bool es)

  Enable of serial read-out (otherwise parallel read-out)
- bool GetEnableSerial () const
- void SetEnableJTAGReadout (bool jr)

  Enable of read-out via JTAG.
- bool GetEnableJTAGReadout () const
- void SetEdgeResolution (const EdgeResolution r)
- EdgeResolution GetEdgeResolution () const
- void SetMaxEventSize (int sz)

  Set the maximum number of hits per event.
- uint8_t GetMaxEventSize () const

  Extract the maximum number of hits per event.
- void SetRejectFIFOFull (bool rej=true)

  Reject hits when readout FIFO full.
- bool GetRejectFIFOFull () const

  Are hits rejected when readout FIFO is full?
- void SetEnableReadoutOccupancy (const bool ro=true)

  Enable the readout of buffer occupancies for each event (for debugging purposes)
- bool GetEnableReadoutOccupancy () const
- void SetEnableReadoutSeparator (const bool ro=true)

  Enable the readout of separators for each event (for debugging purposes, valid if readout of occupancies is enabled)
- bool GetEnableReadoutSeparator () const
- void SetTriggerCountOffset (uint16_t tco)

  Set offset for the trigger time tag counter.
- uint16_t GetTriggerCountOffset () const

  Extract trigger time tag count offset.
- void SetChannelOffset (int channel, uint16_t offset)
- uint16_t GetChannelOffset (int channel) const
- void SetAllChannelsOffset (uint16_t offset)
- void SetCoarseCountOffset (uint16_t cco)

  Set offset for the coarse time counter.
- uint16_t GetCoarseCountOffset () const

  Extract offset for the coarse time counter.
- void SetDLLAdjustment (int tap, uint8_t adj)

  Set the DLL taps adjustments with a resolution of $\sim$10 ps.
- uint8_t GetDLLAdjustment (int tap) const
- void SetAllTapsDLLAdjustment (uint8_t adj)
- void SetRCAdjustment (int tap, uint8_t adj)
- uint8_t GetRCAdjustment (int tap)
- void SetWidthResolution (const WidthResolution r)
- WidthResolution GetWidthResolution () const
- void SetVernierOffset (const uint8_t vo)

  Set the offset in vernier decoding.
- uint8_t GetVernierOffset () const

  Extract the offset in vernier decoding.
- void SetDeadTime (const DeadTime dt)
- DeadTime GetDeadTime () const
- void SetLeadingMode (const bool lead=true)

Enable the detection of leading edges.

- bool GetLeadingMode () const

  Extract the status for the detection of leading edges.
- void SetTrailingMode (const bool trail=true)

  Enable/disable the detection of trailing edges.
- bool GetTrailingMode () const

  Extract the status for the detection of trailing edges.
- void SetTriggerMatchingMode (const bool trig=true)

  Set the enable status of trigger matching mode.
- bool GetTriggerMatchingMode () const

  Extract the enable status of trigger matching mode.
- void SetEdgesPairing (const bool pair=true)

  Enable the pairing of leading and trailing edges (overrides individual enable of leading/trailing edges)
- bool GetEdgesPairing () const
- void SetSetupParity (const bool sp=true)

  Set the parity of setup data (should be an even parity)
- bool GetSetupParity () const

  Extract the parity of setup data (should be an even parity)
- void SetConstantValues ()

  Ensure that the critical constant values are properly set in the setup word.
- uint16_t GetTriggerLatency () const

  Effective trigger latency in number of clock cycles (when no counter roll-over is used)
- void Dump (int verb=1, std::ostream &os=std::cout) const

5.11.1 Detailed Description

Setup word to be sent to the HPTDC chip.

Object handling the configuration word provided by/to the HPTDC chip

Author

Laurent Forthomme `laurent.forthomme@cern.ch`

Date

16 Apr 2015

5.11.2 Constructor & Destructor Documentation

5.11.2.1 TDCConfiguration::TDCConfiguration ( )

5.11.2.2 virtual TDCConfiguration::~TDCConfiguration ( ) `[inline]`, `[virtual]`

5.11.3 Member Function Documentation

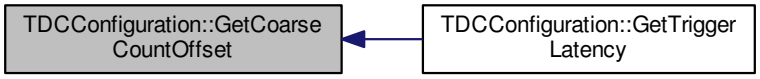
5.11.3.1 void TDCConfiguration::Dump (int *verb* = `1`, std::ostream & *os* = `std::cout`) const

5.11.3.2 uint16_t TDCConfiguration::GetChannelOffset (int *channel*) const `[inline]`

5.11.3.3 uint16_t TDCConfiguration::GetCoarseCountOffset ( ) const `[inline]`

Extract offset for the coarse time counter.

Here is the caller graph for this function:



5.11.3.4 DeadTime TDCConfiguration::GetDeadTime ( ) const [inline]

5.11.3.5 uint8_t TDCConfiguration::GetDLLAdjustment (int *tap*) const [inline]

5.11.3.6 EdgeResolution TDCConfiguration::GetEdgeResolution ( ) const [inline]

5.11.3.7 bool TDCConfiguration::GetEdgesPairing ( ) const [inline]

5.11.3.8 uint16_t TDCConfiguration::GetEnableError ( ) const [inline]

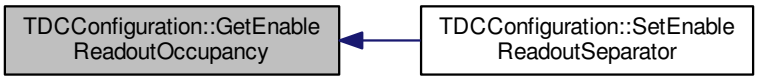
5.11.3.9 bool TDCConfiguration::GetEnableErrorBypass ( ) const [inline]

5.11.3.10 bool TDCConfiguration::GetEnableErrorMark ( ) const [inline]

5.11.3.11 bool TDCConfiguration::GetEnableJTAGReadout ( ) const [inline]

5.11.3.12 bool TDCConfiguration::GetEnableReadoutOccupancy ( ) const [inline]

Here is the caller graph for this function:



5.11.3.13 bool TDCConfiguration::GetEnableReadoutSeparator ( ) const [inline]

5.11.3.14 bool TDCConfiguration::GetEnableSerial ( ) const [inline]

5.11.3.15 bool TDCConfiguration::GetLeadingMode ( ) const [inline]

Extract the status for the detection of leading edges.

5.11.3.16 uint8_t TDCConfiguration::GetMaxEventSize ( ) const [inline]

Extract the maximum number of hits per event.

5.11.3.17 uint8_t TDCConfiguration::GetNumWords ( ) const [inline]

Number of words in the configuration.

Return the number of words making up the full configuration word.

5.11.3.18 uint8_t TDCConfiguration::GetRCAdjustment (int *tap*) [inline]

5.11.3.19 bool TDCConfiguration::GetRejectFIFOFull ( ) const [inline]

Are hits rejected when readout FIFO is full?

Extract whether or not hits are rejected once FIFO is full.

5.11.3.20 bool TDCConfiguration::GetSetupParity ( ) const [inline]

Extract the parity of setup data (should be an even parity)

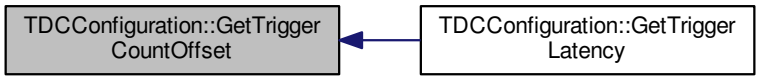
5.11.3.21 bool TDCConfiguration::GetTrailingMode ( ) const [inline]

Extract the status for the detection of trailing edges.

5.11.3.22 uint16_t TDCConfiguration::GetTriggerCountOffset ( ) const [inline]

Extract trigger time tag count offset.

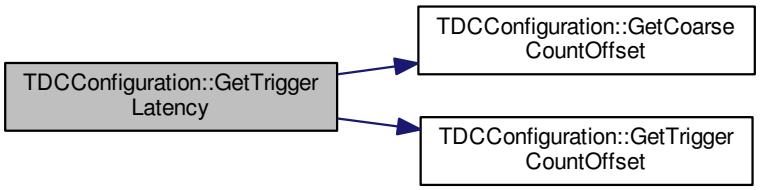
Here is the caller graph for this function:



5.11.3.23 uint16_t TDCConfiguration::GetTriggerLatency ( ) const [inline]

Effective trigger latency in number of clock cycles (when no counter roll-over is used)

Here is the call graph for this function:



5.11.3.24 bool TDCConfiguration::GetTriggerMatchingMode ( ) const [inline]

Extract the enable status of trigger matching mode.

5.11.3.25 uint8_t TDCConfiguration::GetVernierOffset ( ) const [inline]

Extract the offset in vernier decoding.

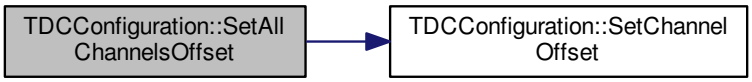
5.11.3.26 WidthResolution TDCConfiguration::GetWidthResolution ( ) const [inline]

5.11.3.27 word_t TDCConfiguration::GetWord (const unsigned int *i*) const [inline]

Retrieve one subset from the setup word.

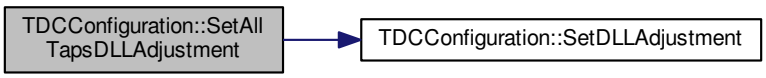
5.11.3.28 void TDCConfiguration::SetAllChannelsOffset (uint16_t *offset*) [inline]

Here is the call graph for this function:



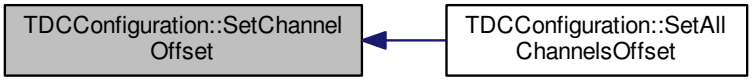
5.11.3.29 void TDCConfiguration::SetAllTapsDLLAdjustment (uint8_t *adj*) [inline]

Here is the call graph for this function:



5.11.3.30 void TDCConfiguration::SetChannelOffset (int *channel,* uint16_t *offset*) [inline]

Here is the caller graph for this function:



5.11.3.31 void TDCConfiguration::SetCoarseCountOffset (uint16_t *cco*) [inline]

Set offset for the coarse time counter.

5.11.3.32 void TDCConfiguration::SetConstantValues ( ) [inline]

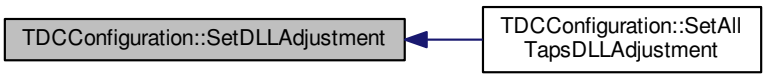
Ensure that the critical constant values are properly set in the setup word.

5.11.3.33 void TDCConfiguration::SetDeadTime (const DeadTime *dt*) [inline]

5.11.3.34 void TDCConfiguration::SetDLLAdjustment (int *tap,* uint8_t *adj*) [inline]

Set the DLL taps adjustments with a resolution of ∼10 ps.

Here is the caller graph for this function:



5.11.3.35 void TDCConfiguration::SetEdgeResolution (const EdgeResolution *r*) [inline]

5.11.3.36 void TDCConfiguration::SetEdgesPairing (const bool *pair* = true) [inline]

Enable the pairing of leading and trailing edges (overrides individual enable of leading/trailing edges)

5.11.3.37 void TDCConfiguration::SetEnableError (const uint16_t & *err*) [inline]

Enable internal error types for generation of global error signals.

5.11.3.38 void TDCConfiguration::SetEnableErrorBypass (bool *eb*) [inline]

Bypass TDC chip if global error signal is set.

5.11.3.39 void TDCConfiguration::SetEnableErrorMark (bool *em*) [inline]

Mark events with error if global error signal is set.

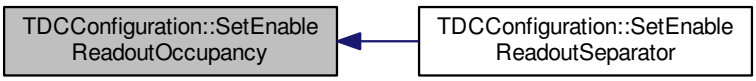
5.11.3.40 void TDCConfiguration::SetEnableJTAGReadout (bool *jr*) [inline]

Enable of read-out via JTAG.

5.11.3.41 void TDCConfiguration::SetEnableReadoutOccupancy (const bool *ro* = true) [inline]

Enable the readout of buffer occupancies for each event (for debugging purposes)

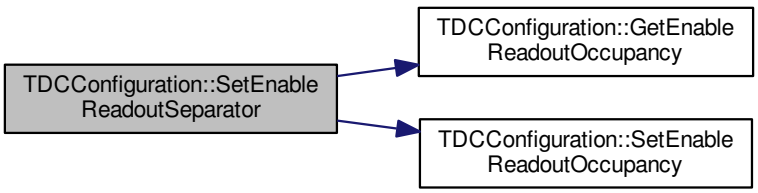
Here is the caller graph for this function:



5.11.3.42 void TDCConfiguration::SetEnableReadoutSeparator (const bool *ro* = true) [inline]

Enable the readout of separators for each event (for debugging purposes, valid if readout of occupancies is enabled)

Here is the call graph for this function:



5.11.3.43 void TDCConfiguration::SetEnableSerial (bool *es*) [inline]

Enable of serial read-out (otherwise parallel read-out)

5.11.3.44 void TDCConfiguration::SetLeadingMode (const bool *lead* = true) [inline]

Enable the detection of leading edges.

5.11.3.45 void TDCConfiguration::SetMaxEventSize (int *sz*) [inline]

Set the maximum number of hits per event.

Set the maximum number of hits that can be recorded for each event. It is always rounded to the next power of 2 (in the range 0-128), and if lower than 0 or bigger than 128 then set to unimited.

5.11.3.46 void TDCConfiguration::SetRCAdjustment (int *tap,* uint8_t *adj*) [inline]

5.11.3.47 void TDCConfiguration::SetRejectFIFOFull (bool *rej* = true) [inline]

Reject hits when readout FIFO full.

Set whether or not hits are rejected once FIFO is full.

5.11.3.48 void TDCConfiguration::SetSetupParity (const bool *sp* = true) [inline]

Set the parity of setup data (should be an even parity)

5.11.3.49 void TDCConfiguration::SetTrailingMode (const bool *trail* = true) [inline]

Enable/disable the detection of trailing edges.

5.11.3.50 void TDCConfiguration::SetTriggerCountOffset (uint16_t *tco*) [inline]

Set offset for the trigger time tag counter.

5.11.3.51 void TDCConfiguration::SetTriggerMatchingMode (const bool *trig* = true) [inline]

Set the enable status of trigger matching mode.

5.11.3.52 void TDCConfiguration::SetVernierOffset (const uint8_t *vo*) [inline]

Set the offset in vernier decoding.

5.11.3.53 void TDCConfiguration::SetWidthResolution (const WidthResolution *r*) [inline]

5.11.3.54 void TDCConfiguration::SetWord (const unsigned int *i,* const word_t *word*) [inline]

Set one bit(s) subset in the setup word.

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

- include/TDCConfiguration.h

5.12 TDCEvent Class Reference

HPTDC event parser.

`#include <TDCEvent.h>`

Public Types

- enum EventType {
  Invalid =-1, GroupHeader =0, GroupTrailer, TDCHeader,
  TDCTrailer, LeadingEdge, TrailingEdge, Error,
  Debug }

Public Member Functions

- TDCEvent (const uint32_t &word)
- virtual ~TDCEvent ()
- EventType GetType () const

  Type of packet read out from the TDC.
- unsigned int GetTDCId () const

  Programmed identifier of master TDC.
- uint16_t GetEventId () const

  Event identifier from event counter.
- uint16_t GetWordCount () const

  Total number of words in event (including headers and trailers)
- uint16_t GetBunchId () const

  Bunch identifier of trigger (or trigger time tag)
- uint32_t GetLeadingTime (bool pair=false) const

  Leading edge measurement in programmed time resolution.
- uint8_t GetWidth () const

  Width of pulse in programmed time resolution.
- uint32_t GetTrailingTime () const

  Trailing edge measurement in programmed time resolution.
- uint16_t GetErrorFlags () const

  Return error flags if an error condition has been detected.

5.12.1 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

20 Apr 2015

5.12.2 Constructor & Destructor Documentation

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

5.12.2.2 virtual TDCEvent::~TDCEvent ( ) [inline], [virtual]

5.12.3 Member Function Documentation

5.12.3.1 uint16_t TDCEvent::GetBunchId ( ) const [inline]

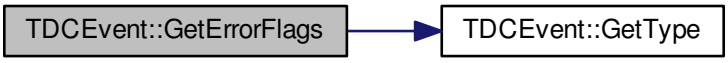
Bunch identifier of trigger (or trigger time tag)

Here is the call graph for this function:

5.12.3.2 uint16_t TDCEvent::GetErrorFlags ( ) const [inline]

Return error flags if an error condition has been detected.

Here is the call graph for this function:



5.12.3.3 uint16_t TDCEvent::GetEventId ( ) const [inline]

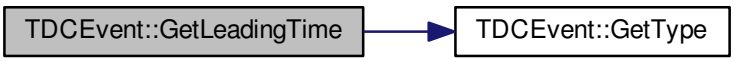
Event identifier from event counter.

Here is the call graph for this function:

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

Leading edge measurement in programmed time resolution.

Here is the call graph for this function:



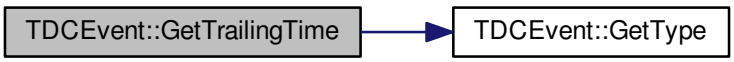
5.12.3.5 unsigned int TDCEvent::GetTDCId ( ) const [inline]

Programmed identifier of master TDC.

5.12.3.6 uint32_t TDCEvent::GetTrailingTime ( ) const [inline]

Trailing edge measurement in programmed time resolution.

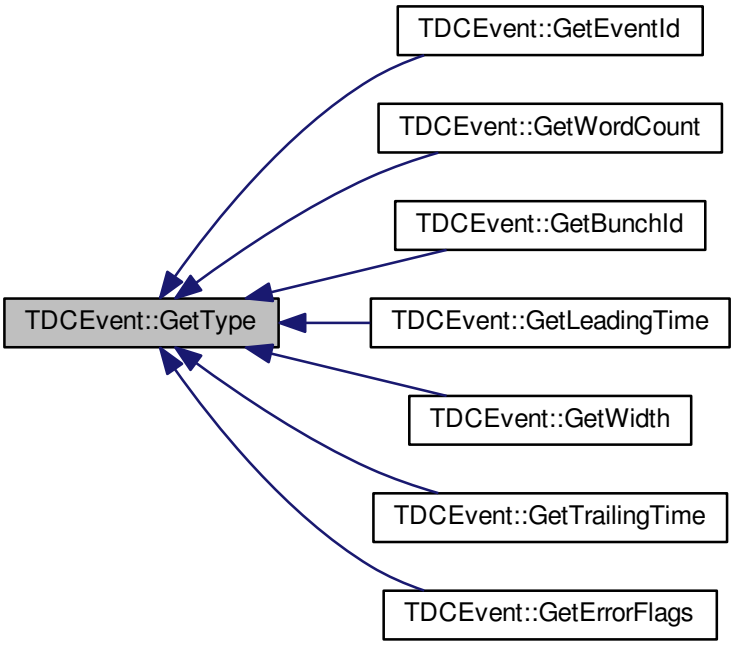
Here is the call graph for this function:



5.12.3.7 EventType TDCEvent::GetType ( ) const [inline]

Type of packet read out from the TDC.

Here is the caller graph for this function:



5.12.3.8 uint8_t TDCEvent::GetWidth ( ) const [inline]

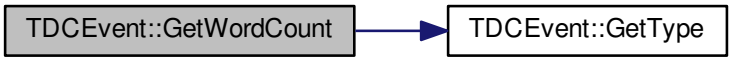
Width of pulse in programmed time resolution.

Here is the call graph for this function:

5.12.3.9 uint16_t TDCEvent::GetWordCount ( ) const [inline]

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

Here is the call graph for this function:



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

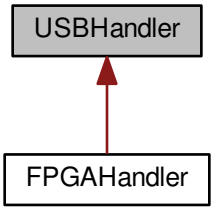
- include/TDCEvent.h

5.13 USBHandler Class Reference

Generic USB communication handler.

`#include <USBHandler.h>`

Inheritance diagram for USBHandler:



Public Member Functions

- USBHandler (const char *dev)
- virtual ~USBHandler ()
- void Init ()
- void DumpDevice (libusb_device *dev, int verb=1, std::ostream &out=std::cout)

Protected Member Functions

- void Write (uint32_t word, uint8_t size) const

  Write a word to the USB device.
- uint32_t Fetch (uint8_t size) const

  Receive a word from the USB device.

5.13.1 Detailed Description

Generic USB communication handler.

Date

21 Apr 2015

Author

Laurent Forthomme laurent.forthomme@cern.ch

5.13.2 Constructor & Destructor Documentation

5.13.2.1 USBHandler::USBHandler (const char * *dev*)

5.13.2.2 virtual USBHandler::~USBHandler ( ) [inline], [virtual]

5.13.3 Member Function Documentation

5.13.3.1 void USBHandler::DumpDevice (libusb_device * *dev*, int *verb* = 1, std::ostream & *out* = std::cout)

5.13.3.2 uint32_t USBHandler::Fetch (uint8_t *size*) const [inline], [protected]

Receive a word from the USB device.

5.13.3.3 void USBHandler::Init ( )

5.13.3.4 void USBHandler::Write (uint32_t *word*, uint8_t *size*) const [inline], [protected]

Write a word to the USB device.

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

- include/USBHandler.h

Index

~Client
Client, 15
~Exception
Exception, 16
~FPGAHandler
FPGAHandler, 22
~Message
Message, 27
~Messenger
Messenger, 29
~Socket
Socket, 32
~SocketMessage
SocketMessage, 39
~TDCConfiguration
TDCConfiguration, 44
~TDCEvent
TDCEvent, 51
~USBHandler
USBHandler, 56

AcceptConnections
Socket, 32

Bind
Socket, 33
Broadcast
Messenger, 29

CLIENT
Socket communication objects, 7
ChannelSelectError
HPTDC chip control, 10
Client, 13
~Client, 15
Client, 15
Connect, 15
Disconnect, 15
GetType, 15
ParseMessage, 15
Receive, 15
Send, 15
CloseFile
FPGAHandler, 22
CoarseError
HPTDC chip control, 10
config
file_header_t, 20
Connect
Client, 15
Messenger, 30
ControlParityError
HPTDC chip control, 10

DETECTOR
Socket communication objects, 8
DT_100ns
HPTDC chip control, 9
DT_10ns
HPTDC chip control, 9
DT_30ns
HPTDC chip control, 9
DT_5ns
HPTDC chip control, 9
DeadTime
HPTDC chip control, 9
Debug
HPTDC chip control, 10
Decode
HTTPMessage, 24
Description
Exception, 17
Disconnect
Client, 15
Messenger, 30
Dump
Exception, 17
HTTPMessage, 25
Message, 27
SocketMessage, 39
TDCConfiguration, 44
DumpConnected
Socket, 33
DumpDevice
USBHandler, 56

E_100ps
HPTDC chip control, 10
E_12p5ns
HPTDC chip control, 10
E_1p6ns
HPTDC chip control, 10
E_200ps
HPTDC chip control, 10
E_3p12ns
HPTDC chip control, 10
E_400ps
HPTDC chip control, 10
E_6p25ns
HPTDC chip control, 10

E_800ps
HPTDC chip control, 10
EdgeResolution
HPTDC chip control, 9
EnabledError
HPTDC chip control, 10
Encode
HTTPMessage, 25
Error
HPTDC chip control, 10
ErrorNumber
Exception, 17
EventType
HPTDC chip control, 10
Exception, 16
∼Exception, 16
Description, 17
Dump, 17
ErrorNumber, 17
Exception, 16
From, 17
Type, 18
TypeString, 18

fBuffer
Socket, 35
fMaster
Socket, 35
FPGAHandler, 20
∼FPGAHandler, 22
CloseFile, 22
FPGAHandler, 22
GetConfiguration, 22
GetFilename, 22
GetType, 22
OpenFile, 22
ReadBuffer, 22
SetConfiguration, 22
fPort
Socket, 35
fReadFds
Socket, 35
fSocketsConnected
Socket, 35
fString
Message, 28
Fetch
USBHandler, 56
FetchMessage
Socket, 33
file_header_t, 19
config, 20
magic, 20
run_id, 20
spill_id, 20
From
Exception, 17

GetBunchId
TDCEvent, 51
GetChannelOffset
TDCConfiguration, 44
GetCoarseCountOffset
TDCConfiguration, 44
GetConfiguration
FPGAHandler, 22
GetDLLAdjustment
TDCConfiguration, 45
GetDeadTime
TDCConfiguration, 45
GetEdgeResolution
TDCConfiguration, 45
GetEdgesPairing
TDCConfiguration, 45
GetEnableError
TDCConfiguration, 45
GetEnableErrorBypass
TDCConfiguration, 45
GetEnableErrorMark
TDCConfiguration, 45
GetEnableJTAGReadout
TDCConfiguration, 45
GetEnableReadoutOccupancy
TDCConfiguration, 45
GetEnableReadoutSeparator
TDCConfiguration, 45
GetEnableSerial
TDCConfiguration, 45
GetErrorFlags
TDCEvent, 52
GetEventId
TDCEvent, 52
GetFilename
FPGAHandler, 22
GetIntValue
SocketMessage, 39
GetKey
HTTPMessage, 25
Message, 27
SocketMessage, 39
GetLeadingMode
TDCConfiguration, 45
GetLeadingTime
TDCEvent, 52
GetMaxEventSize
TDCConfiguration, 45
GetNumWords
TDCConfiguration, 45
GetPort
Socket, 33
GetRCAdjustment
TDCConfiguration, 46
GetRejectFIFOFull
TDCConfiguration, 46
GetSetupParity
TDCConfiguration, 46
GetSocketId

Socket, 33
GetSocketType
Socket, 33
GetString
Message, 27
SocketMessage, 40
GetTDCId
TDCEvent, 53
GetTrailingMode
TDCConfiguration, 46
GetTrailingTime
TDCEvent, 53
GetTriggerCountOffset
TDCConfiguration, 46
GetTriggerLatency
TDCConfiguration, 46
GetTriggerMatchingMode
TDCConfiguration, 47
GetType
Client, 15
FPGAHandler, 22
Messenger, 30
TDCEvent, 53
GetValue
SocketMessage, 40
GetVectorValue
SocketMessage, 40
GetVernierOffset
TDCConfiguration, 47
GetWidth
TDCEvent, 54
GetWidthResolution
TDCConfiguration, 47
GetWord
TDCConfiguration, 47
GetWordCount
TDCEvent, 54
GroupHeader
HPTDC chip control, 10
GroupTrailer
HPTDC chip control, 10

HPTDC chip control, 9
ChannelSelectError, 10
CoarseError, 10
ControlParityError, 10
DT_100ns, 9
DT_10ns, 9
DT_30ns, 9
DT_5ns, 9
DeadTime, 9
Debug, 10
E_100ps, 10
E_12p5ns, 10
E_1p6ns, 10
E_200ps, 10
E_3p12ns, 10
E_400ps, 10
E_6p25ns, 10
E_800ps, 10
EdgeResolution, 9
EnabledError, 10
Error, 10
EventType, 10
GroupHeader, 10
GroupTrailer, 10
Invalid, 10
JTAGInstructionParityError, 10
L1BufferParityError, 10
LeadingEdge, 10
ReadoutFIFOParityError, 10
ReadoutStateError, 10
SetupParityError, 10
TDCHeader, 10
TDCTrailer, 10
TrailingEdge, 10
TriggerFIFOParityError, 10
TriggerMatchingError, 10
VernierError, 10
W_100ns, 11
W_100ps, 11
W_12p5ns, 11
W_1p6ns, 11
W_200ns, 11
W_200ps, 11
W_25ns, 11
W_3p2ns, 11
W_400ns, 11
W_400ps, 11
W_50ns, 11
W_6p25ns, 11
W_800ns, 11
W_800ps, 11
WidthResolution, 10
HTTPMessage, 22
Decode, 24
Dump, 25
Encode, 25
GetKey, 25
HTTPMessage, 24

INVALID
Socket communication objects, 7
Init
USBHandler, 56
Invalid
HPTDC chip control, 10
IsFromWeb
Message, 27
IsWebSocket
Socket, 33

JTAGInstructionParityError
HPTDC chip control, 10

L1BufferParityError
HPTDC chip control, 10
LeadingEdge

HPTDC chip control, 10
Listen
Socket, 34
ListenerInfo, 25
name, 26
type, 26

MASTER
Socket communication objects, 7
magic
file_header_t, 20
Message, 26
∼Message, 27
Dump, 27
fString, 28
GetKey, 27
GetString, 27
IsFromWeb, 27
Message, 27
Messenger, 28
∼Messenger, 29
Broadcast, 29
Connect, 30
Disconnect, 30
GetType, 30
Messenger, 29
Receive, 30
Send, 30

name
ListenerInfo, 26

OpenFile
FPGAHandler, 22

ParseMessage
Client, 15
PrepareConnection
Socket, 34

ReadBuffer
FPGAHandler, 22
ReadoutFIFOParityError
HPTDC chip control, 10
ReadoutStateError
HPTDC chip control, 10
Receive
Client, 15
Messenger, 30
run_id
file_header_t, 20

SelectConnections
Socket, 34
Send
Client, 15
Messenger, 30
SendMessage
Socket, 34
SetAllChannelsOffset
TDCConfiguration, 47
SetAllTapsDLLAdjustment
TDCConfiguration, 47
SetChannelOffset
TDCConfiguration, 47
SetCoarseCountOffset
TDCConfiguration, 48
SetConfiguration
FPGAHandler, 22
SetConstantValues
TDCConfiguration, 48
SetDLLAdjustment
TDCConfiguration, 48
SetDeadTime
TDCConfiguration, 48
SetEdgeResolution
TDCConfiguration, 48
SetEdgesPairing
TDCConfiguration, 48
SetEnableError
TDCConfiguration, 48
SetEnableErrorBypass
TDCConfiguration, 48
SetEnableErrorMark
TDCConfiguration, 48
SetEnableJTAGReadout
TDCConfiguration, 49
SetEnableReadoutOccupancy
TDCConfiguration, 49
SetEnableReadoutSeparator
TDCConfiguration, 49
SetEnableSerial
TDCConfiguration, 49
SetKeyValue
SocketMessage, 40, 41
SetLeadingMode
TDCConfiguration, 49
SetMaxEventSize
TDCConfiguration, 49
SetPort
Socket, 34
SetRCAdjustment
TDCConfiguration, 50
SetRejectFIFOFull
TDCConfiguration, 50
SetSetupParity
TDCConfiguration, 50
SetSocketId
Socket, 34
SetTrailingMode
TDCConfiguration, 50
SetTriggerCountOffset
TDCConfiguration, 50
SetTriggerMatchingMode
TDCConfiguration, 50
SetVernierOffset
TDCConfiguration, 50
SetWidthResolution

TDCConfiguration, 50
SetWord
TDCConfiguration, 50
SetupParityError
HPTDC chip control, 10
Socket, 30
~Socket, 32
AcceptConnections, 32
Bind, 33
DumpConnected, 33
fBuffer, 35
fMaster, 35
fPort, 35
fReadFds, 35
fSocketsConnected, 35
FetchMessage, 33
GetPort, 33
GetSocketId, 33
GetSocketType, 33
IsWebSocket, 33
Listen, 34
PrepareConnection, 34
SelectConnections, 34
SendMessage, 34
SetPort, 34
SetSocketId, 34
Socket, 32
SocketCollection, 32
Start, 34
Stop, 35
Socket communication objects, 7
CLIENT, 7
DETECTOR, 8
INVALID, 7
MASTER, 7
SocketType, 7
WEBSOCKET_CLIENT, 7
SocketCollection
Socket, 32
SocketMessage, 35
~SocketMessage, 39
Dump, 39
GetIntValue, 39
GetKey, 39
GetString, 40
GetValue, 40
GetVectorValue, 40
SetKeyValue, 40, 41
SocketMessage, 37–39
SocketType
Socket communication objects, 7
spill_id
file_header_t, 20
Start
Socket, 34
Stop
Socket, 35

TDCConfiguration, 42
~TDCConfiguration, 44
Dump, 44
GetChannelOffset, 44
GetCoarseCountOffset, 44
GetDLLAdjustment, 45
GetDeadTime, 45
GetEdgeResolution, 45
GetEdgesPairing, 45
GetEnableError, 45
GetEnableErrorBypass, 45
GetEnableErrorMark, 45
GetEnableJTAGReadout, 45
GetEnableReadoutOccupancy, 45
GetEnableReadoutSeparator, 45
GetEnableSerial, 45
GetLeadingMode, 45
GetMaxEventSize, 45
GetNumWords, 45
GetRCAdjustment, 46
GetRejectFIFOFull, 46
GetSetupParity, 46
GetTrailingMode, 46
GetTriggerCountOffset, 46
GetTriggerLatency, 46
GetTriggerMatchingMode, 47
GetVernierOffset, 47
GetWidthResolution, 47
GetWord, 47
SetAllChannelsOffset, 47
SetAllTapsDLLAdjustment, 47
SetChannelOffset, 47
SetCoarseCountOffset, 48
SetConstantValues, 48
SetDLLAdjustment, 48
SetDeadTime, 48
SetEdgeResolution, 48
SetEdgesPairing, 48
SetEnableError, 48
SetEnableErrorBypass, 48
SetEnableErrorMark, 48
SetEnableJTAGReadout, 49
SetEnableReadoutOccupancy, 49
SetEnableReadoutSeparator, 49
SetEnableSerial, 49
SetLeadingMode, 49
SetMaxEventSize, 49
SetRCAdjustment, 50
SetRejectFIFOFull, 50
SetSetupParity, 50
SetTrailingMode, 50
SetTriggerCountOffset, 50
SetTriggerMatchingMode, 50
SetVernierOffset, 50
SetWidthResolution, 50
SetWord, 50
TDCConfiguration, 44
TDCEvent, 50
~TDCEvent, 51

GetBunchId, 51
GetErrorFlags, 52
GetEventId, 52
GetLeadingTime, 52
GetTDCId, 53
GetTrailingTime, 53
GetType, 53
GetWidth, 54
GetWordCount, 54
TDCEvent, 51
TDCHeader
HPTDC chip control, 10
TDCTrailer
HPTDC chip control, 10
TrailingEdge
HPTDC chip control, 10
TriggerFIFOParityError
HPTDC chip control, 10
TriggerMatchingError
HPTDC chip control, 10
Type
Exception, 18
type
ListenerInfo, 26
TypeString
Exception, 18

USBHandler, 55
∼USBHandler, 56
DumpDevice, 56
Fetch, 56
Init, 56
USBHandler, 56
Write, 56

VernierError
HPTDC chip control, 10

W_100ns
HPTDC chip control, 11
W_100ps
HPTDC chip control, 11
W_12p5ns
HPTDC chip control, 11
W_1p6ns
HPTDC chip control, 11
W_200ns
HPTDC chip control, 11
W_200ps
HPTDC chip control, 11
W_25ns
HPTDC chip control, 11
W_3p2ns
HPTDC chip control, 11
W_400ns
HPTDC chip control, 11
W_400ps
HPTDC chip control, 11
W_50ns
HPTDC chip control, 11
W_6p25ns
HPTDC chip control, 11
W_800ns
HPTDC chip control, 11
W_800ps
HPTDC chip control, 11
WEBSOCKET_CLIENT
Socket communication objects, 7
WidthResolution
HPTDC chip control, 10
Write
USBHandler, 56