

2015 Test beam Run Control

Generated by Doxygen 1.8.9.1

Thu Apr 23 2015 18:13:43

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 | 10 |
| 4.2.2.1 | CoreClockSource | 10 |
| 4.2.2.2 | DeadTime | 10 |
| 4.2.2.3 | DLLClockSource | 10 |
| 4.2.2.4 | DLLSpeedMode | 10 |
| 4.2.2.5 | EdgeResolution | 10 |
| 4.2.2.6 | EnabledError | 11 |
| 4.2.2.7 | EventType | 11 |
| 4.2.2.8 | IOClockSource | 11 |
| 4.2.2.9 | SerialClockSource | 11 |
| 4.2.2.10 | WidthResolution | 12 |
| 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 | 16 |
| 5.1.3.4 | ParseMessage | 16 |
| 5.1.3.5 | Receive | 17 |
| 5.1.3.6 | Send | 17 |
| 5.2 | Exception Class Reference | 18 |
| 5.2.1 | Detailed Description | 18 |
| 5.2.2 | Constructor & Destructor Documentation | 18 |
| 5.2.2.1 | Exception | 18 |
| 5.2.2.2 | Exception | 19 |
| 5.2.2.3 | ~Exception | 19 |
| 5.2.3 | Member Function Documentation | 19 |
| 5.2.3.1 | Description | 19 |
| 5.2.3.2 | Dump | 20 |
| 5.2.3.3 | ErrorNumber | 21 |
| 5.2.3.4 | From | 21 |
| 5.2.3.5 | Type | 21 |
| 5.2.3.6 | TypeString | 22 |
| 5.3 | file_header_t Struct Reference | 22 |
| 5.3.1 | Detailed Description | 23 |
| 5.3.2 | Field Documentation | 23 |
| 5.3.2.1 | config | 23 |
| 5.3.2.2 | magic | 23 |
| 5.3.2.3 | run_id | 23 |
| 5.3.2.4 | spill_id | 23 |
| 5.4 | FPGAHandler Class Reference | 24 |
| 5.4.1 | Detailed Description | 25 |
| 5.4.2 | Constructor & Destructor Documentation | 25 |
| 5.4.2.1 | FPGAHandler | 25 |
| 5.4.2.2 | ~FPGAHandler | 25 |
| 5.4.3 | Member Function Documentation | 26 |
| 5.4.3.1 | CloseFile | 26 |
| 5.4.3.2 | GetConfiguration | 26 |
| 5.4.3.3 | GetFilename | 26 |
| 5.4.3.4 | GetType | 26 |
| 5.4.3.5 | OpenFile | 26 |

| | | |
|---------|--|----|
| 5.4.3.6 | ReadBuffer | 26 |
| 5.4.3.7 | SetConfiguration | 26 |
| 5.5 | HTTPMessage Class Reference | 26 |
| 5.5.1 | Detailed Description | 27 |
| 5.5.2 | Constructor & Destructor Documentation | 28 |
| 5.5.2.1 | HTTPMessage | 28 |
| 5.5.2.2 | HTTPMessage | 28 |
| 5.5.3 | Member Function Documentation | 28 |
| 5.5.3.1 | Decode | 29 |
| 5.5.3.2 | Dump | 29 |
| 5.5.3.3 | Encode | 29 |
| 5.5.3.4 | GetKey | 29 |
| 5.6 | ListenerInfo Struct Reference | 29 |
| 5.6.1 | Detailed Description | 29 |
| 5.6.2 | Field Documentation | 30 |
| 5.6.2.1 | name | 30 |
| 5.6.2.2 | type | 30 |
| 5.7 | Message Class Reference | 30 |
| 5.7.1 | Detailed Description | 31 |
| 5.7.2 | Constructor & Destructor Documentation | 31 |
| 5.7.2.1 | Message | 31 |
| 5.7.2.2 | Message | 31 |
| 5.7.2.3 | Message | 31 |
| 5.7.2.4 | ~Message | 31 |
| 5.7.3 | Member Function Documentation | 31 |
| 5.7.3.1 | Dump | 31 |
| 5.7.3.2 | GetKey | 31 |
| 5.7.3.3 | GetString | 31 |
| 5.7.3.4 | IsFromWeb | 32 |
| 5.7.4 | Field Documentation | 32 |
| 5.7.4.1 | fString | 32 |
| 5.8 | Messenger Class Reference | 32 |
| 5.8.1 | Detailed Description | 33 |
| 5.8.2 | Constructor & Destructor Documentation | 34 |
| 5.8.2.1 | Messenger | 34 |
| 5.8.2.2 | Messenger | 34 |
| 5.8.2.3 | ~Messenger | 34 |
| 5.8.3 | Member Function Documentation | 34 |
| 5.8.3.1 | Broadcast | 34 |
| 5.8.3.2 | Connect | 35 |

| | | |
|----------|--|----|
| 5.8.3.3 | Disconnect | 35 |
| 5.8.3.4 | GetType | 36 |
| 5.8.3.5 | Receive | 36 |
| 5.8.3.6 | Send | 37 |
| 5.9 | Socket Class Reference | 38 |
| 5.9.1 | Detailed Description | 40 |
| 5.9.2 | Member Typedef Documentation | 40 |
| 5.9.2.1 | SocketCollection | 40 |
| 5.9.3 | Constructor & Destructor Documentation | 40 |
| 5.9.3.1 | Socket | 40 |
| 5.9.3.2 | Socket | 40 |
| 5.9.3.3 | ~Socket | 40 |
| 5.9.4 | Member Function Documentation | 40 |
| 5.9.4.1 | AcceptConnections | 40 |
| 5.9.4.2 | Bind | 41 |
| 5.9.4.3 | DumpConnected | 41 |
| 5.9.4.4 | FetchMessage | 41 |
| 5.9.4.5 | GetPort | 42 |
| 5.9.4.6 | GetSocketId | 42 |
| 5.9.4.7 | GetSocketType | 43 |
| 5.9.4.8 | IsWebSocket | 43 |
| 5.9.4.9 | Listen | 43 |
| 5.9.4.10 | PrepareConnection | 44 |
| 5.9.4.11 | SelectConnections | 44 |
| 5.9.4.12 | SendMessage | 45 |
| 5.9.4.13 | SetPort | 45 |
| 5.9.4.14 | SetSocketId | 45 |
| 5.9.4.15 | Start | 45 |
| 5.9.4.16 | Stop | 46 |
| 5.9.5 | Field Documentation | 47 |
| 5.9.5.1 | fBuffer | 47 |
| 5.9.5.2 | fMaster | 47 |
| 5.9.5.3 | fPort | 47 |
| 5.9.5.4 | fReadFds | 47 |
| 5.9.5.5 | fSocketsConnected | 47 |
| 5.10 | SocketMessage Class Reference | 47 |
| 5.10.1 | Detailed Description | 49 |
| 5.10.2 | Constructor & Destructor Documentation | 49 |
| 5.10.2.1 | SocketMessage | 49 |
| 5.10.2.2 | SocketMessage | 49 |

| | | |
|-----------|--|----|
| 5.10.2.3 | SocketMessage | 49 |
| 5.10.2.4 | SocketMessage | 49 |
| 5.10.2.5 | SocketMessage | 49 |
| 5.10.2.6 | SocketMessage | 49 |
| 5.10.2.7 | SocketMessage | 50 |
| 5.10.2.8 | SocketMessage | 50 |
| 5.10.2.9 | SocketMessage | 50 |
| 5.10.2.10 | SocketMessage | 51 |
| 5.10.2.11 | SocketMessage | 51 |
| 5.10.2.12 | ~SocketMessage | 51 |
| 5.10.3 | Member Function Documentation | 51 |
| 5.10.3.1 | Dump | 51 |
| 5.10.3.2 | GetIntValue | 51 |
| 5.10.3.3 | GetKey | 51 |
| 5.10.3.4 | GetString | 52 |
| 5.10.3.5 | GetValue | 52 |
| 5.10.3.6 | GetVectorValue | 52 |
| 5.10.3.7 | SetKeyValue | 53 |
| 5.10.3.8 | SetKeyValue | 53 |
| 5.10.3.9 | SetKeyValue | 53 |
| 5.10.3.10 | SetKeyValue | 54 |
| 5.11 | TDCConfiguration Class Reference | 54 |
| 5.11.1 | Detailed Description | 57 |
| 5.11.2 | Constructor & Destructor Documentation | 57 |
| 5.11.2.1 | TDCConfiguration | 57 |
| 5.11.2.2 | TDCConfiguration | 57 |
| 5.11.2.3 | ~TDCConfiguration | 57 |
| 5.11.3 | Member Function Documentation | 57 |
| 5.11.3.1 | Dump | 58 |
| 5.11.3.2 | GetChannelOffset | 59 |
| 5.11.3.3 | GetCoarseCountOffset | 59 |
| 5.11.3.4 | GetDeadTime | 59 |
| 5.11.3.5 | GetDLLAdjustment | 60 |
| 5.11.3.6 | GetEdgeResolution | 60 |
| 5.11.3.7 | GetEdgesPairing | 60 |
| 5.11.3.8 | GetEnableError | 61 |
| 5.11.3.9 | GetEnableErrorBypass | 61 |
| 5.11.3.10 | GetEnableErrorMark | 61 |
| 5.11.3.11 | GetEnableJTAGReadout | 61 |
| 5.11.3.12 | GetEnableReadoutOccupancy | 61 |

| | |
|---|----|
| 5.11.3.13 GetEnableReadoutSeparator | 61 |
| 5.11.3.14 GetEnableSerial | 61 |
| 5.11.3.15 GetLeadingMode | 61 |
| 5.11.3.16 GetMaxEventSize | 62 |
| 5.11.3.17 GetNumWords | 62 |
| 5.11.3.18 GetRCAdjustment | 62 |
| 5.11.3.19 GetRejectFIFOFull | 62 |
| 5.11.3.20 GetSetupParity | 62 |
| 5.11.3.21 GetTestInvert | 62 |
| 5.11.3.22 GetTestMode | 62 |
| 5.11.3.23 GetTrailingMode | 62 |
| 5.11.3.24 GetTriggerCountOffset | 63 |
| 5.11.3.25 GetTriggerLatency | 63 |
| 5.11.3.26 GetTriggerMatchingMode | 63 |
| 5.11.3.27 GetVernierOffset | 64 |
| 5.11.3.28 GetWidthResolution | 64 |
| 5.11.3.29 GetWord | 64 |
| 5.11.3.30 SetAllChannelsOffset | 64 |
| 5.11.3.31 SetAllTapsDLLAdjustment | 65 |
| 5.11.3.32 SetChannelOffset | 65 |
| 5.11.3.33 SetCoarseCountOffset | 65 |
| 5.11.3.34 SetConstantValues | 65 |
| 5.11.3.35 SetDeadTime | 66 |
| 5.11.3.36 SetDLLAdjustment | 66 |
| 5.11.3.37 SetEdgeResolution | 66 |
| 5.11.3.38 SetEdgesPairing | 66 |
| 5.11.3.39 SetEnableError | 66 |
| 5.11.3.40 SetEnableErrorBypass | 66 |
| 5.11.3.41 SetEnableErrorMark | 67 |
| 5.11.3.42 SetEnableJTAGReadout | 67 |
| 5.11.3.43 SetEnableReadoutOccupancy | 67 |
| 5.11.3.44 SetEnableReadoutSeparator | 67 |
| 5.11.3.45 SetEnableSerial | 67 |
| 5.11.3.46 SetLeadingMode | 67 |
| 5.11.3.47 SetMaxEventSize | 68 |
| 5.11.3.48 SetRCAdjustment | 68 |
| 5.11.3.49 SetRejectFIFOFull | 68 |
| 5.11.3.50 SetSetupParity | 68 |
| 5.11.3.51 SetTestInvert | 68 |
| 5.11.3.52 SetTestMode | 68 |

| | |
|---|----|
| 5.11.3.53 SetTrailingMode | 68 |
| 5.11.3.54 SetTriggerCountOffset | 68 |
| 5.11.3.55 SetTriggerMatchingMode | 69 |
| 5.11.3.56 SetVernierOffset | 69 |
| 5.11.3.57 SetWidthResolution | 69 |
| 5.11.3.58 SetWord | 69 |
| 5.12 TDCEvent Class Reference | 69 |
| 5.12.1 Detailed Description | 70 |
| 5.12.2 Constructor & Destructor Documentation | 70 |
| 5.12.2.1 TDCEvent | 70 |
| 5.12.2.2 ~TDCEvent | 70 |
| 5.12.3 Member Function Documentation | 70 |
| 5.12.3.1 GetBunchId | 70 |
| 5.12.3.2 GetErrorFlags | 70 |
| 5.12.3.3 GetEventId | 71 |
| 5.12.3.4 GetLeadingTime | 71 |
| 5.12.3.5 GetTDCId | 71 |
| 5.12.3.6 GetTrailingTime | 71 |
| 5.12.3.7 GetType | 72 |
| 5.12.3.8 GetWidth | 72 |
| 5.12.3.9 GetWordCount | 72 |
| 5.13 USBHandler Class Reference | 73 |
| 5.13.1 Detailed Description | 74 |
| 5.13.2 Constructor & Destructor Documentation | 74 |
| 5.13.2.1 USBHandler | 74 |
| 5.13.2.2 ~USBHandler | 74 |
| 5.13.3 Member Function Documentation | 74 |
| 5.13.3.1 DumpDevice | 74 |
| 5.13.3.2 Fetch | 74 |
| 5.13.3.3 Init | 74 |
| 5.13.3.4 Write | 75 |

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 | 18 |
| file_header_t | 22 |
| ListenerInfo | 29 |
| Message | 30 |
| HTTPMessage | 26 |
| SocketMessage | 47 |
| Socket | 38 |
| Client | 13 |
| FPGAHandler | 24 |
| Messenger | 32 |
| TDCCConfiguration | 54 |
| TDCEvent | 69 |
| USBHandler | 73 |
| FPGAHandler | 24 |

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 | 18 |
| file_header_t | Header to the output files | 22 |
| FPGAHandler | Driver for timing detectors' FPGA readout | 24 |
| HTTPMessage | Message to be transmitted through a WebSocket protocol | 26 |
| ListenerInfo | Information on a socket's listener | 29 |
| Message | Base socket message type | 30 |
| Messenger | Base master object for the socket | 32 |
| Socket | Base socket object from which clients/master from a socket inherit | 38 |
| SocketMessage | Socket-passed message type | 47 |
| TDCCConfiguration | Setup word to be sent to the HPTDC chip | 54 |
| TDCEvent | HPTDC event parser | 69 |
| USBHandler | Generic USB communication handler | 73 |

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 `TDCConfiguration::DLLSpeedMode` { `TDCConfiguration::DLL_40MHz =0x0, TDCConfiguration::DLL_160MHz =0x1, TDCConfiguration::DLL_320MHz =0x2, TDCConfiguration::DLL_Illegal =0x3 }`
- enum `TDCConfiguration::SerialClockSource` { `TDCConfiguration::Serial_pll_clock_80 =0x0, TDCConfiguration::Serial_pll_clock_160 =0x1, TDCConfiguration::Serial_pll_clock_40 =0x2, TDCConfiguration::Serial_aux_clock =0x3 }`
- enum `TDCConfiguration::IOClockSource` { `TDCConfiguration::IO_clock_40 =0x0, TDCConfiguration::IO_pll_clock_80 =0x1, TDCConfiguration::IO_pll_clock_160 =0x2, TDCConfiguration::IO_aux_clock =0x3 }`
- enum `TDCConfiguration::CoreClockSource` { `TDCConfiguration::Core_clock_40 =0x0, TDCConfiguration::Core_pll_clock_80 =0x1, TDCConfiguration::Core_pll_clock_160 =0x2, TDCConfiguration::Core_aux_clock =0x3 }`
- enum `TDCConfiguration::DLLClockSource` {
`TDCConfiguration::DLL_clock_40 =0x0, TDCConfiguration::DLL_pll_clock_40 =0x1, TDCConfiguration::DLL_pll_clock_160 =0x2, TDCConfiguration::DLL_pll_clock_320 =0x3,`
`TDCConfiguration::DLL_aux_clock =0x4 }`
- 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::CoreClockSource

Enumerator

Core_clock_40
Core_pll_clock_80
Core_pll_clock_160
Core_aux_clock

4.2.2.2 enum TDCConfiguration::DeadTime

Enumerator

DT_5ns
DT_10ns
DT_30ns
DT_100ns

4.2.2.3 enum TDCConfiguration::DLLClockSource

Enumerator

DLL_clock_40
DLL_pll_clock_40
DLL_pll_clock_160
DLL_pll_clock_320
DLL_aux_clock

4.2.2.4 enum TDCConfiguration::DLLSpeedMode

Enumerator

DLL_40MHz
DLL_160MHz
DLL_320MHz
DLL_Illegal

4.2.2.5 enum TDCConfiguration::EdgeResolution

Enumerator

E_100ps
E_200ps
E_400ps
E_800ps
E_1p6ns
E_3p12ns
E_6p25ns
E_12p5ns

4.2.2.6 enum TDCConfiguration::EnabledError

Enumerator

VernierError
CoarseError
ChannelSelectError
L1BufferParityError
TriggerFIFOParityError
TriggerMatchingError
ReadoutFIFOParityError
ReadoutStateError
SetupParityError
ControlParityError
JTAGInstructionParityError

4.2.2.7 enum TDCEvent::EventType

Enumerator

Invalid
GroupHeader
GroupTrailer
TDCHeader
TDCTrailer
LeadingEdge
TrailingEdge
Error
Debug

4.2.2.8 enum TDCConfiguration::IOClockSource

Enumerator

IO_clock_40
IO_pll_clock_80
IO_pll_clock_160
IO_aux_clock

4.2.2.9 enum TDCConfiguration::SerialClockSource

Enumerator

Serial_pll_clock_80
Serial_pll_clock_160
Serial_pll_clock_40
Serial_aux_clock

4.2.2.10 enum TDCCConfiguration::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

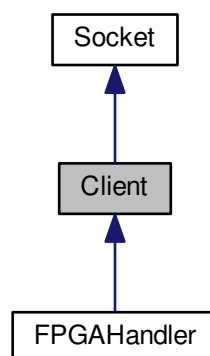
Data Structure Documentation

5.1 Client Class Reference

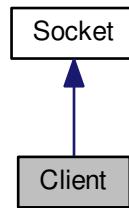
Base client object for the socket.

```
#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]`

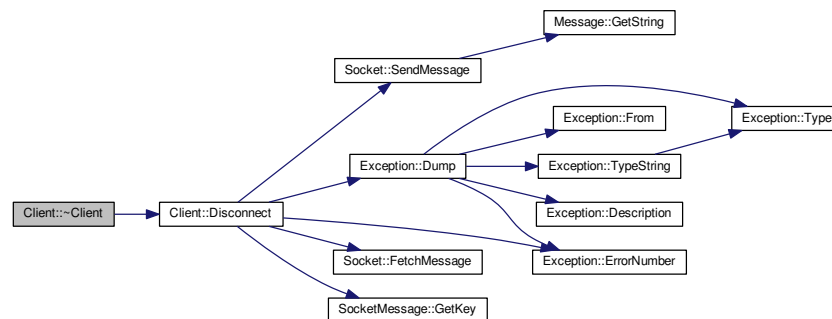
General void client constructor.

5.1.2.2 `Client::Client (int port)`

Bind a socket client to a given port.

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

Here is the call graph for this function:

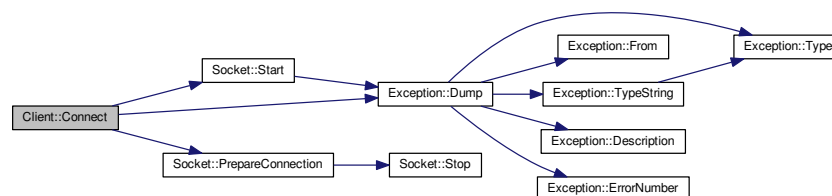


5.1.3 Member Function Documentation

5.1.3.1 `bool Client::Connect ()`

Bind this client to the socket.

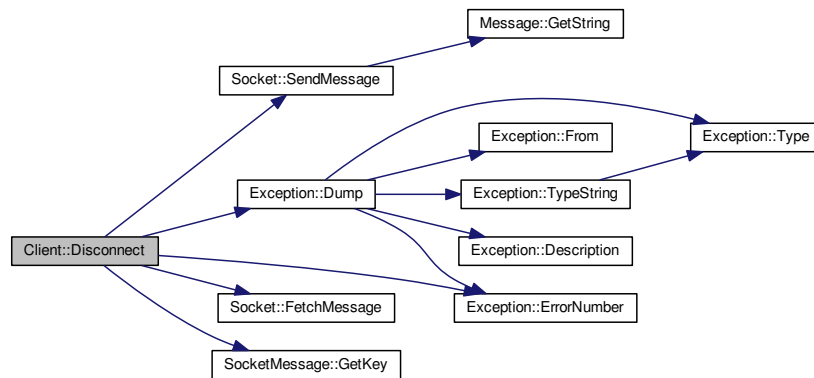
Here is the call graph for this function:



5.1.3.2 `void Client::Disconnect ()`

Unbind this client from the socket.

Here is the call graph for this function:



Here is the caller graph for this function:



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

`Socket` actor type retrieval method.

Reimplemented in `FPGAHandler`.

Here is the caller graph for this function:



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

Parse a `SocketMessage` received from the master.

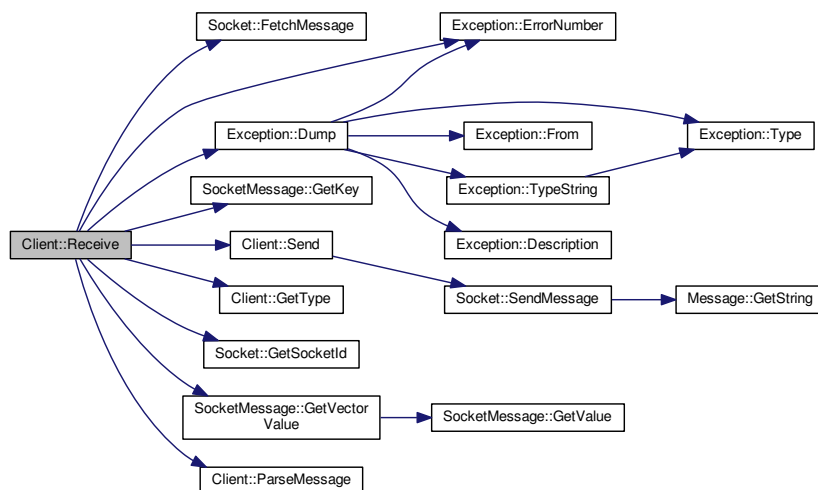
Here is the caller graph for this function:



5.1.3.5 void Client::Receive ()

Receive a socket message from the master.

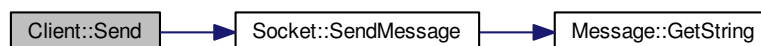
Here is the call graph for this function:



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:



Here is the caller graph for this function:



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

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

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]

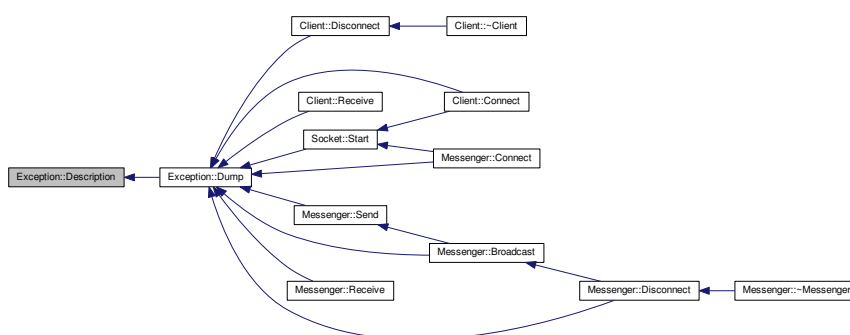
Here is the call graph for this function:



5.2.3 Member Function Documentation

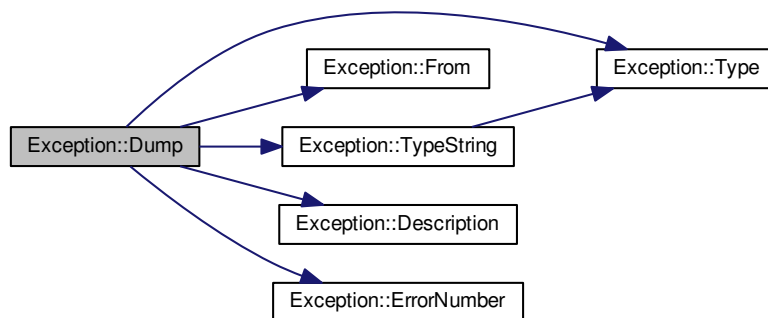
5.2.3.1 `std::string Exception::Description () const` [inline]

Here is the caller graph for this function:

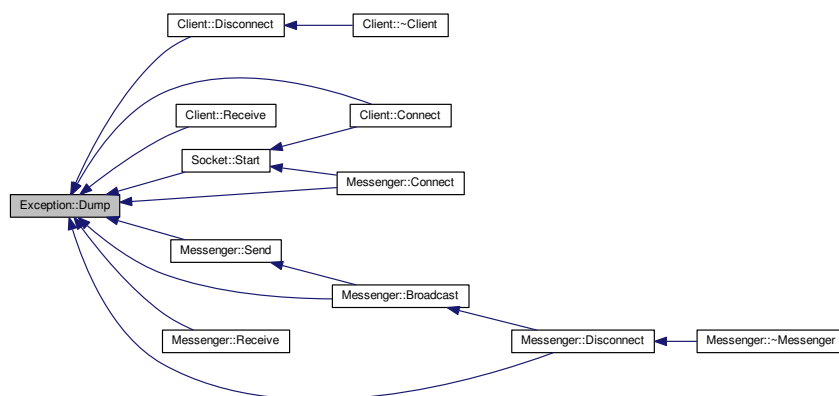


5.2.3.2 `void Exception::Dump (std::ostream & os = std::cerr) const [inline]`

Here is the call graph for this function:

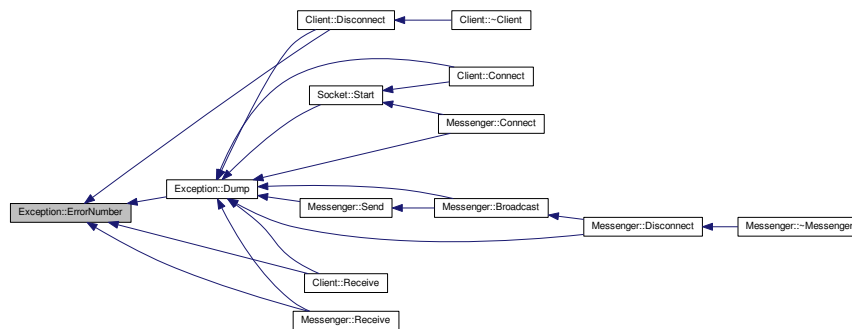


Here is the caller graph for this function:



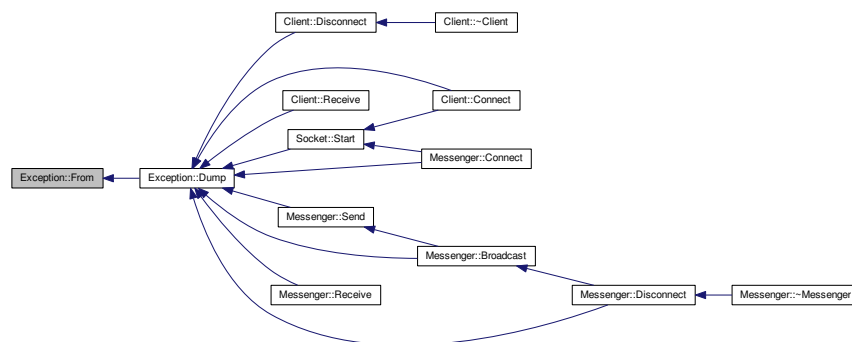
5.2.3.3 `int Exception::ErrorNumber () const [inline]`

Here is the caller graph for this function:



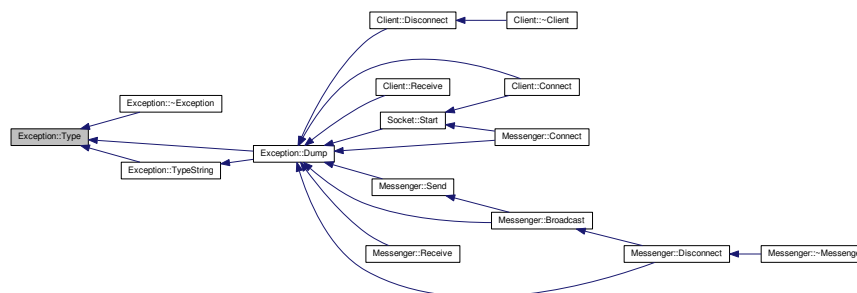
5.2.3.4 `std::string Exception::From () const [inline]`

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:

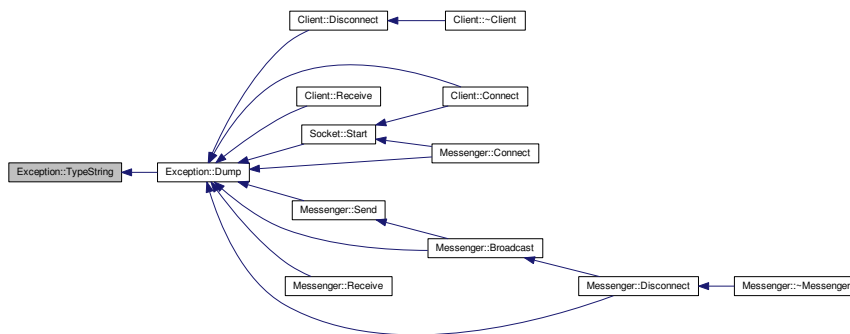


5.2.3.6 `std::string Exception::TypeString () const [inline]`

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:

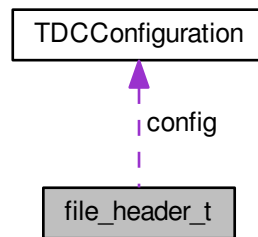
- `include/Exception.h`

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](#)
- [TDCCConfiguration](#) [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 TDCCConfiguration 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:

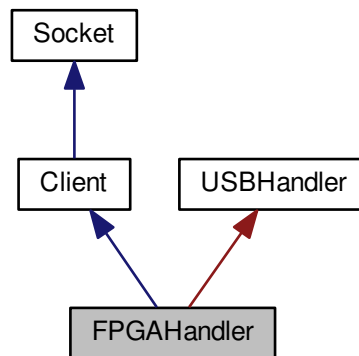
- `include/FPGAHandler.h`

5.4 FPGAHandler Class Reference

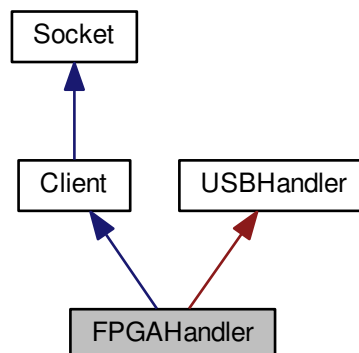
Driver for timing detectors' FPGA readout.

```
#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 [TDCCConfiguration](#) &c)
Submit the HPTDC setup word as a [TDCCConfiguration](#) object.
- [TDCCConfiguration](#) [GetConfiguration](#) ()
Retrieve the HPTDC setup word as a [TDCCConfiguration](#) 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

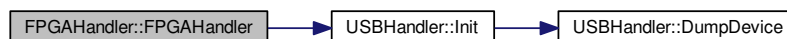
14 Apr 2015

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.

Here is the call graph for this function:



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

Here is the call graph for this function:



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.

Here is the caller graph for this function:



5.4.3.2 TDCCConfiguration FPGAHandler::GetConfiguration () [inline]

Retrieve the HPTDC setup word as a [TDCCConfiguration](#) 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 TDCCConfiguration & c) [inline]

Submit the HPTDC setup word as a [TDCCConfiguration](#) object.

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

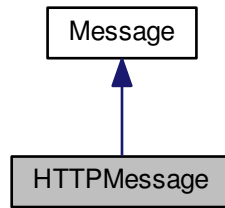
- include/FPGAHandler.h
- src/FPGAHandler.cpp

5.5 HTTPMessage Class Reference

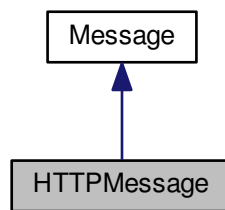
[Message](#) to be transmitted through a WebSocket protocol.

```
#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

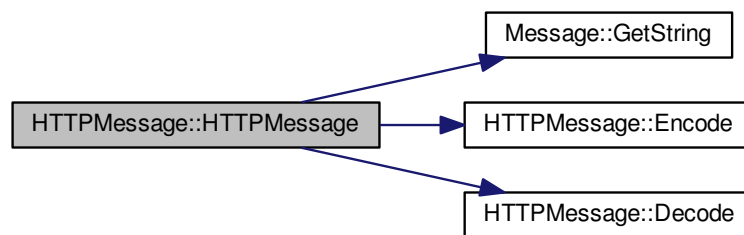
Date

1 Apr 2015

5.5.2 Constructor & Destructor Documentation

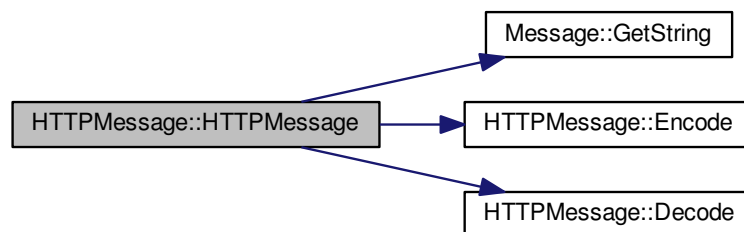
5.5.2.1 HTTPMessage::HTTPMessage (WebSocket * *ws*, Message *m*, MessageAction *a*) [inline]

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]

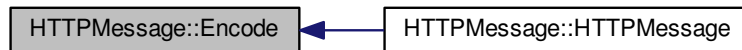
Here is the caller graph for this function:



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:

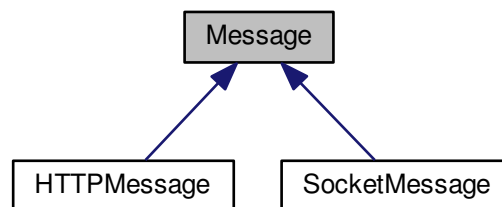
- `include/Messenger.h`

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]`

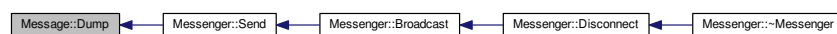
Construct a message from a string.

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]`

Here is the caller graph for this function:



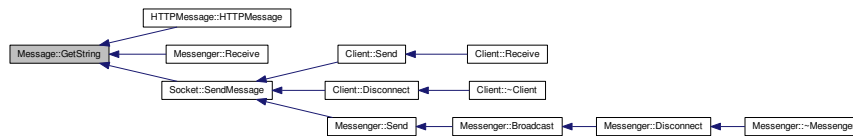
5.7.3.2 `MessageKey Message::GetKey () const` `[inline]`

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:

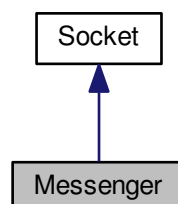
- `include/Message.h`

5.8 Messenger Class Reference

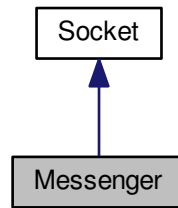
Base master object for the socket.

```
#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

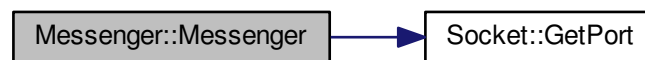
5.8.2.1 Messenger::Messenger ()

Build a void master object or socket actor.

5.8.2.2 Messenger::Messenger (int *port*)

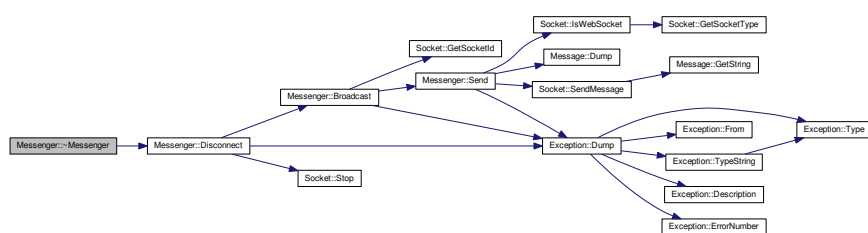
Build a master object to control the socket.

Here is the call graph for this function:



5.8.2.3 Messenger::~~Messenger ()

Here is the call graph for this function:



5.8.3 Member Function Documentation

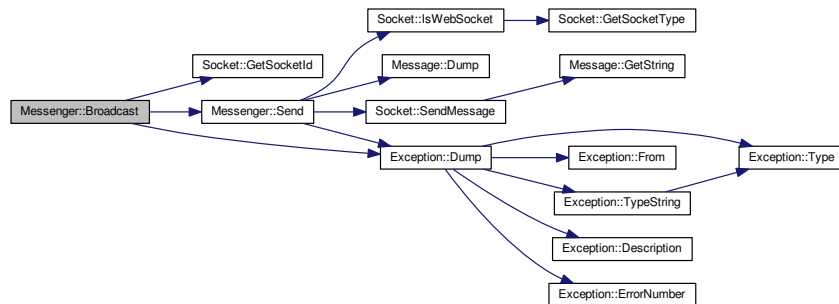
5.8.3.1 void Messenger::Broadcast (const Message & *m*) const

Emit a message to all clients connected through the socket.

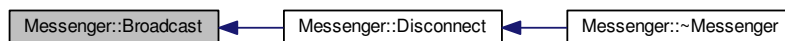
Parameters

| | | |
|----|----------|-------------------------------------|
| in | <i>m</i> | Message to transmit |
|----|----------|-------------------------------------|

Here is the call graph for this function:



Here is the caller graph for this function:

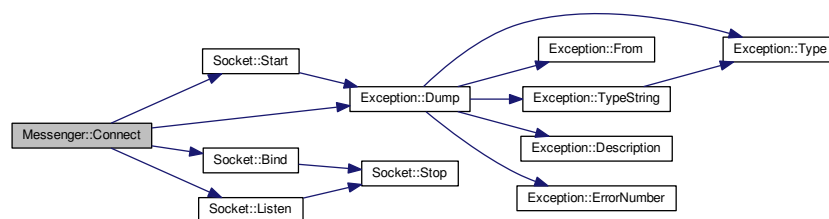


5.8.3.2 bool Messenger::Connect ()

Connect the master to the socket.

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

Here is the call graph for this function:

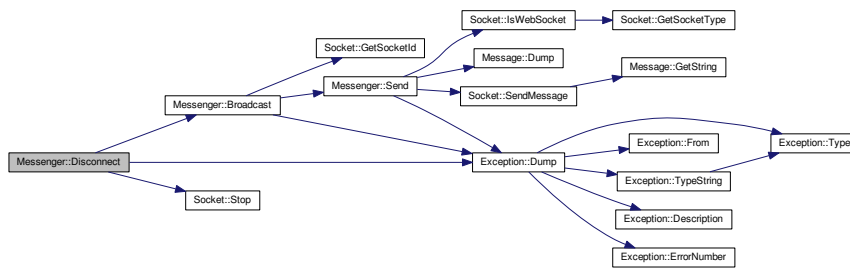


5.8.3.3 void Messenger::Disconnect ()

Remove the master and destroy the socket.

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

Here is the call graph for this function:



Here is the caller graph for this function:



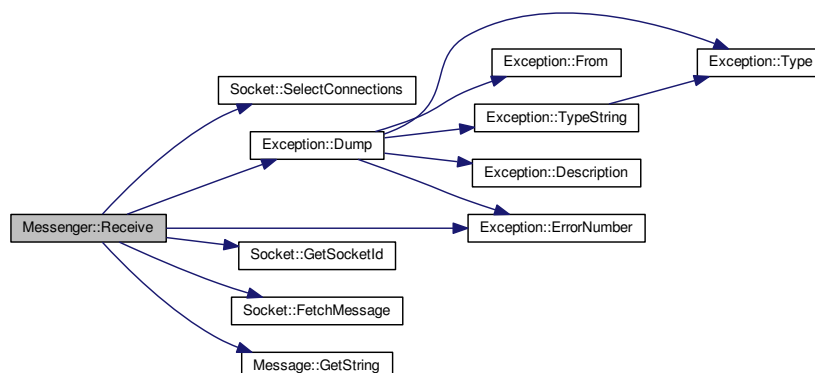
5.8.3.4 SocketType Messenger::GetType () const [inline]

Socket actor type retrieval method.

5.8.3.5 void Messenger::Receive ()

Handle a message reception from a client.

Here is the call graph for this function:



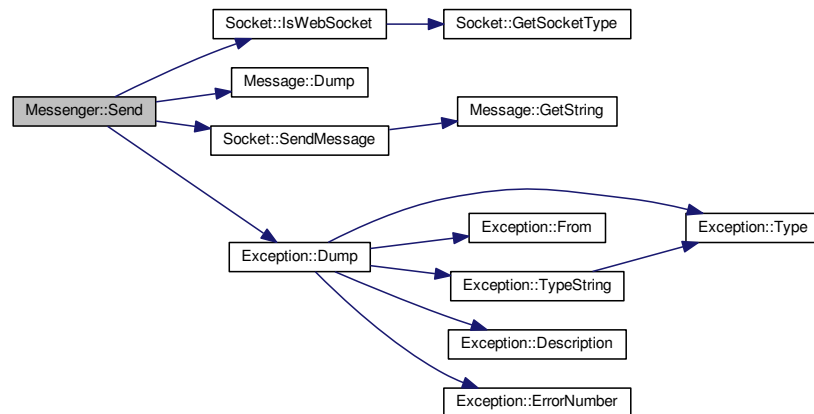
5.8.3.6 `void Messenger::Send (const Message & m, int sid) const` `[inline]`

Send any type of message to any client.

Parameters

| | | |
|----|------------|--|
| in | <i>m</i> | Message to transmit |
| in | <i>sid</i> | Unique identifier of the client on this socket |

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 files:

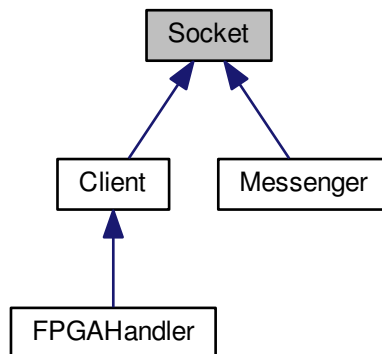
- include/Messenger.h
- src/Messenger.cpp

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

5.9.4 Member Function Documentation

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

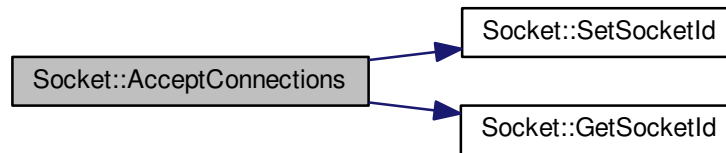
Accept connection from a client.

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

Parameters

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

Here is the call graph for this function:



5.9.4.2 void Socket::Bind () [protected]

Bind a name to a socket.

Returns

Success of the operation

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.4.3 void Socket::DumpConnected () const

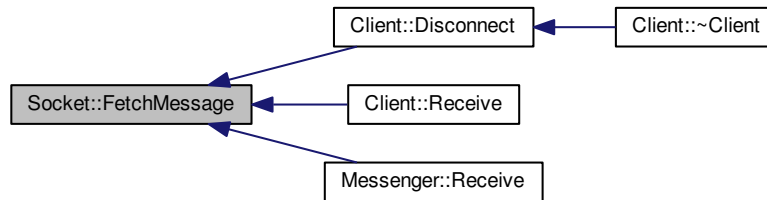
5.9.4.4 Message Socket::FetchMessage (int id = -1) const [protected]

Receive a message from a socket.

Returns

Received message as a `std::string`

Here is the caller graph for this function:



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

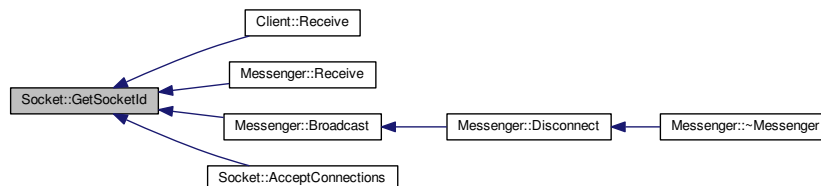
Retrieve the port used for this socket.

Here is the caller graph for this function:



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

Here is the caller graph for this function:



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

Here is the caller graph for this function:

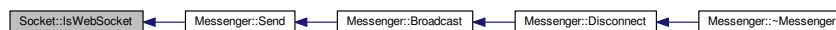


5.9.4.8 bool Socket::IsWebSocket (int *sid*) const [inline]

Here is the call graph for this function:



Here is the caller graph for this function:



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

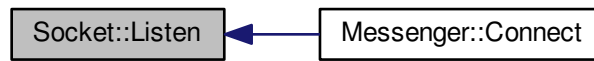
Listen to incoming messages.

Set the socket to listen to any message coming from outside

Here is the call graph for this function:

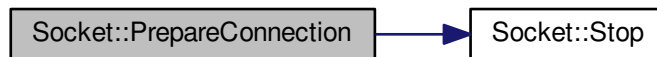


Here is the caller graph for this function:

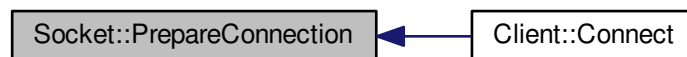


5.9.4.10 void Socket::PrepareConnection () [protected]

Here is the call graph for this function:



Here is the caller graph for this function:



5.9.4.11 void Socket::SelectConnections ()

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

Here is the caller graph for this function:



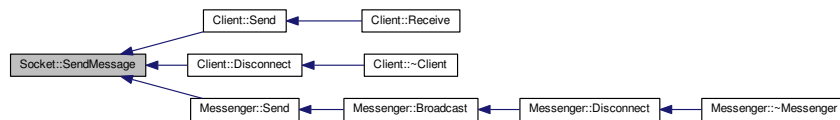
5.9.4.12 `void Socket::SendMessage (Message message, int id = -1) const` [protected]

Send a message on a socket.

Here is the call graph for this function:



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]

Here is the caller graph for this function:



5.9.4.15 `bool Socket::Start ()` [protected]

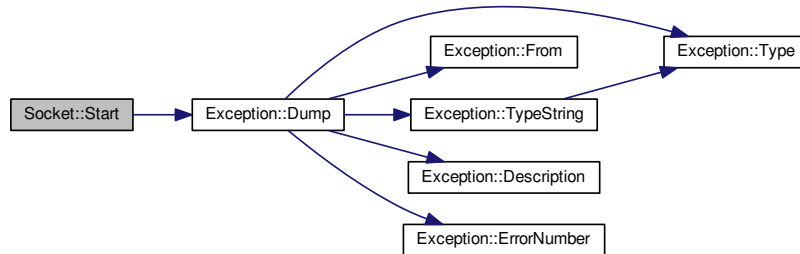
Start the socket.

Launch all mandatory operations to set the socket to be used

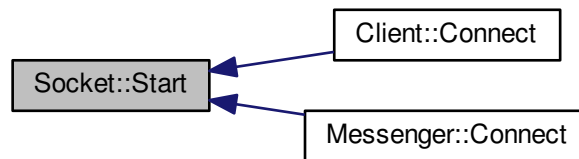
Returns

Success of the operation

Here is the call graph for this function:



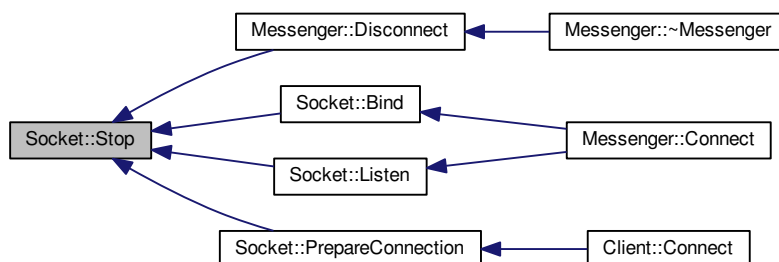
Here is the caller graph for this function:



5.9.4.16 void Socket::Stop ()

Terminates the socket and all attached communications.

Here is the caller graph for this function:



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 files:

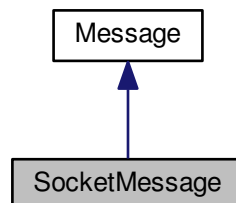
- include/Socket.h
- src/Socket.cpp

5.10 SocketMessage Class Reference

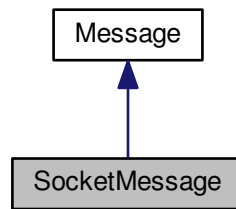
Socket-passed message type.

```
#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]

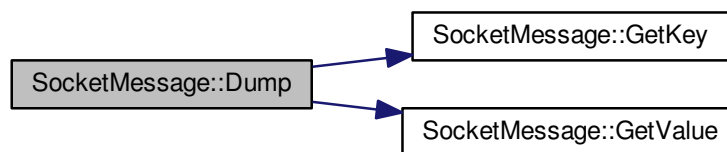
Construct a socket message out of a map of key/string-type value.

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:



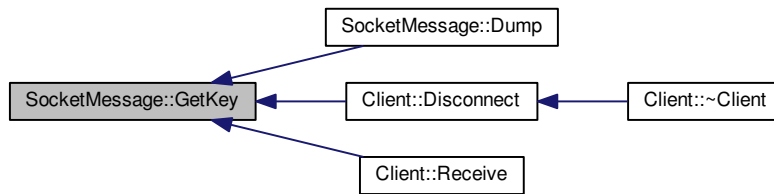
5.10.3.2 int SocketMessage::GetIntValue () const [inline]

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:



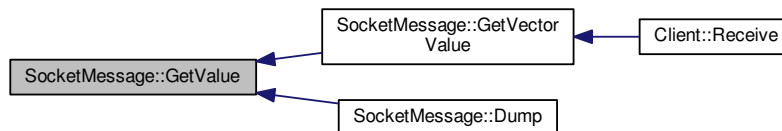
5.10.3.4 `std::string SocketMessage::GetString () const` `[inline]`

Extract the whole key:value message.

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

Extract the message's string value.

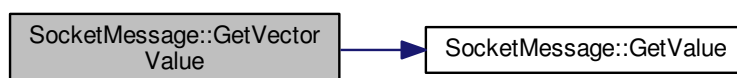
Here is the caller graph for this function:



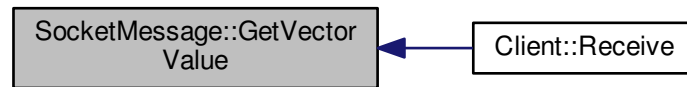
5.10.3.6 `VectorValue SocketMessage::GetVectorValue () const` `[inline]`

Extract the message's vector of string value.

Here is the call graph for this function:



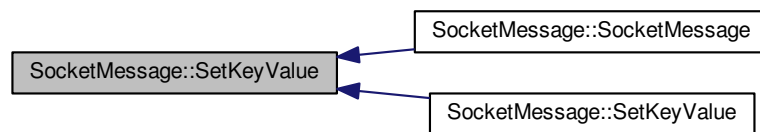
Here is the caller graph for this function:



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, [ReadoutStateError](#) =0x80,
[SetupParityError](#) =0x100, [ControlParityError](#) =0x200, [JTAGInstructionParityError](#) =0x400 }
- enum [DLLSpeedMode](#) { [DLL_40MHz](#) =0x0, [DLL_160MHz](#) =0x1, [DLL_320MHz](#) =0x2, [DLL_Illegal](#) =0x3 }

- enum [SerialClockSource](#) { [Serial_pll_clock_80](#) =0x0, [Serial_pll_clock_160](#) =0x1, [Serial_pll_clock_40](#) =0x2, [Serial_aux_clock](#) =0x3 }
- enum [IOClockSource](#) { [IO_clock_40](#) =0x0, [IO_pll_clock_80](#) =0x1, [IO_pll_clock_160](#) =0x2, [IO_aux_clock](#) =0x3 }
- enum [CoreClockSource](#) { [Core_clock_40](#) =0x0, [Core_pll_clock_80](#) =0x1, [Core_pll_clock_160](#) =0x2, [Core_↔aux_clock](#) =0x3 }
- enum [DLLClockSource](#) { [DLL_clock_40](#) =0x0, [DLL_pll_clock_40](#) =0x1, [DLL_pll_clock_160](#) =0x2, [DLL_pll_clock_320](#) =0x3, [DLL_aux_clock](#) =0x4 }

Public Member Functions

- [TDCConfiguration](#) ()
- [TDCConfiguration](#) (const [TDCConfiguration](#) &c)
- 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 ~ 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)
Channel dead time between hits.
- [DeadTime](#) [GetDeadTime](#) () const
- void [SetTestInvert](#) (const bool ti=true)
Automatic inversion of test pattern. Only used during production testing.
- bool [GetTestInvert](#) () const
- void [SetTestMode](#) (const bool tm=true)
Test mode where hit data are taken from coretest. Only used during production testing.
- bool [GetTestMode](#) () const
- 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 [SetLeadingMode](#) (const bool lead=true)
Enable the detection of leading edges.
- bool [GetLeadingMode](#) () const
Extract the status for the detection of leading 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

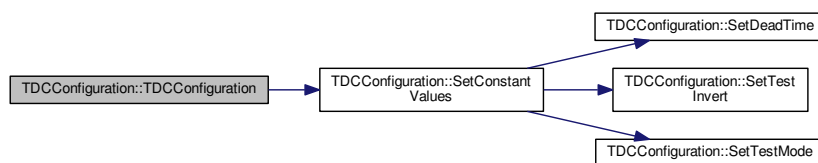
Date

16 Apr 2015

5.11.2 Constructor & Destructor Documentation

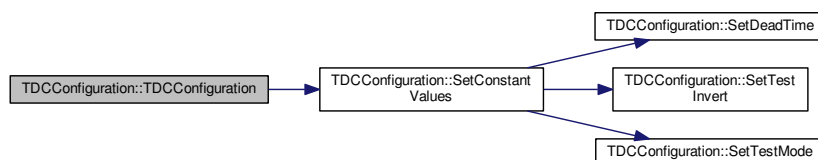
5.11.2.1 TDCCConfiguration::TDCCConfiguration ()

Here is the call graph for this function:



5.11.2.2 TDCCConfiguration::TDCCConfiguration (const TDCCConfiguration & c)

Here is the call graph for this function:

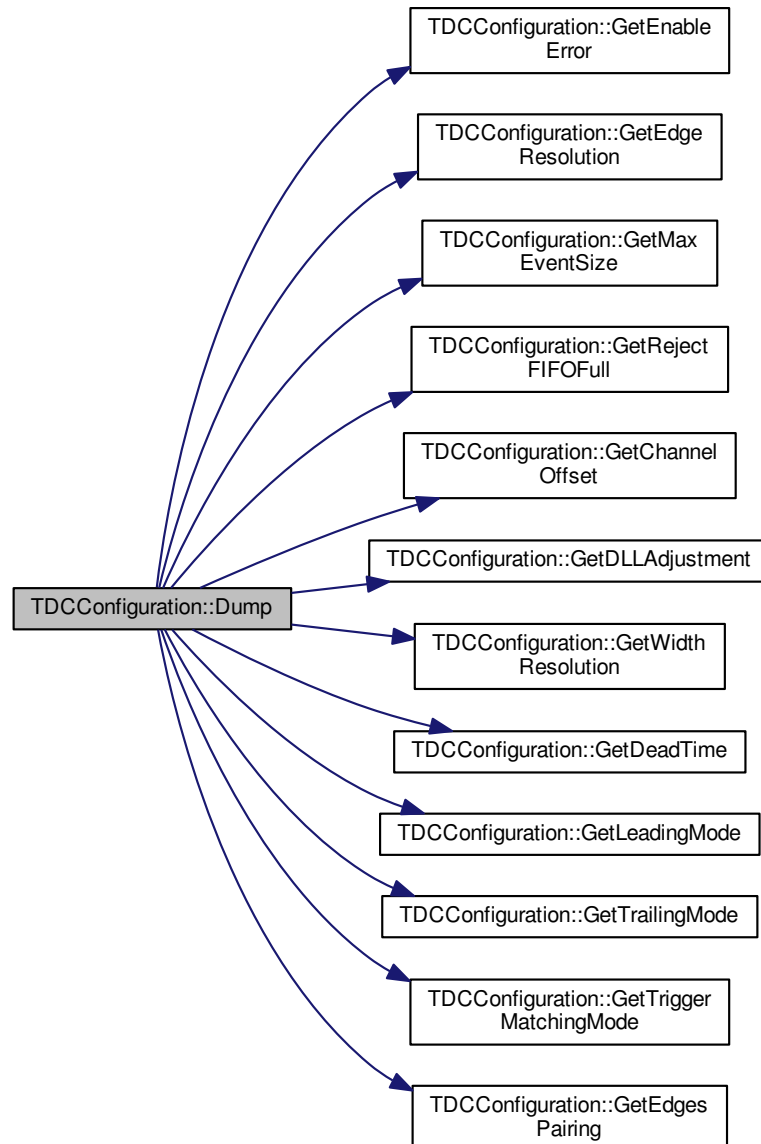


5.11.2.3 virtual TDCCConfiguration::~~TDCCConfiguration () [inline], [virtual]

5.11.3 Member Function Documentation

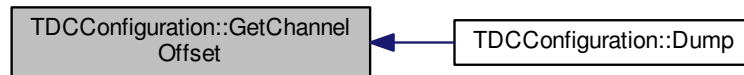
5.11.3.1 `void TDCConfiguration::Dump (int verb = 1, std::ostream & os = std::cout) const`

Here is the call graph for this function:



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

Here is the caller graph for this function:



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

Extract offset for the coarse time counter.

Here is the caller graph for this function:



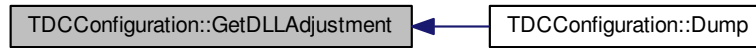
5.11.3.4 `DeadTime TDCCConfiguration::GetDeadTime () const` `[inline]`

Here is the caller graph for this function:



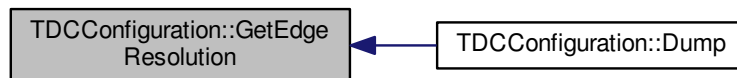
5.11.3.5 `uint8_t TDCConfiguration::GetDLLAdjustment (int tap) const` `[inline]`

Here is the caller graph for this function:



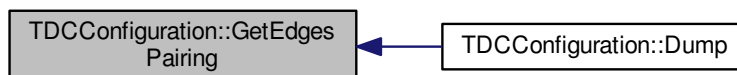
5.11.3.6 `EdgeResolution TDCConfiguration::GetEdgeResolution () const` `[inline]`

Here is the caller graph for this function:



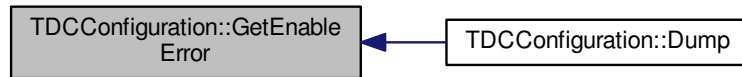
5.11.3.7 `bool TDCConfiguration::GetEdgesPairing () const` `[inline]`

Here is the caller graph for this function:



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

Here is the caller graph for this function:



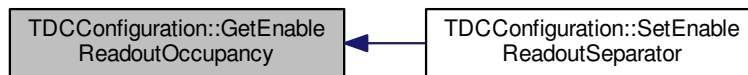
5.11.3.9 `bool TDCConfiguration::GetEnableErrorBypass () const [inline]`

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:



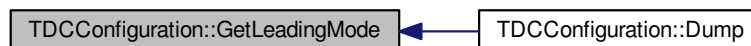
5.11.3.13 `bool TDCConfiguration::GetEnableReadoutSeparator () const [inline]`

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.

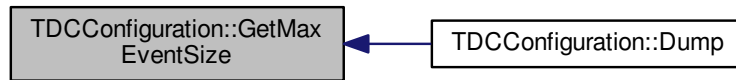
Here is the caller graph for this function:



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

Extract the maximum number of hits per event.

Here is the caller graph for this function:



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.

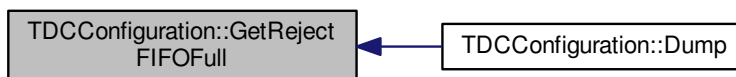
5.11.3.18 `uint8_t TDCConfiguration::GetRCAdjustment (int tap) [inline]`

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.

Here is the caller graph for this function:



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::GetTestInvert () const [inline]`

5.11.3.22 `bool TDCConfiguration::GetTestMode () const [inline]`

5.11.3.23 `bool TDCConfiguration::GetTrailingMode () const [inline]`

Extract the status for the detection of trailing edges.

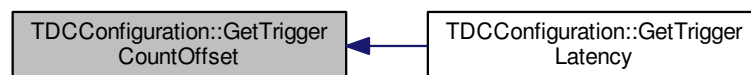
Here is the caller graph for this function:



5.11.3.24 `uint16_t TDCConfiguration::GetTriggerCountOffset () const [inline]`

Extract trigger time tag count offset.

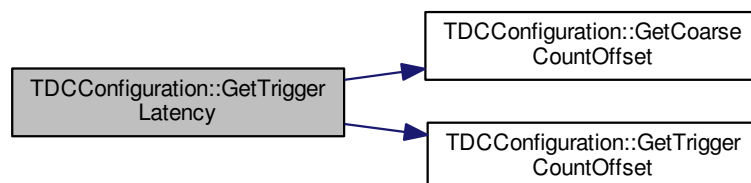
Here is the caller graph for this function:



5.11.3.25 `uint16_t TDCConfiguration::GetTriggerLatency () const [inline]`

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

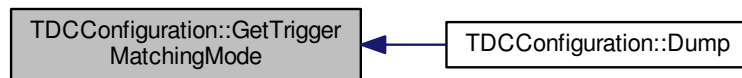
Here is the call graph for this function:



5.11.3.26 `bool TDCConfiguration::GetTriggerMatchingMode () const [inline]`

Extract the enable status of trigger matching mode.

Here is the caller graph for this function:

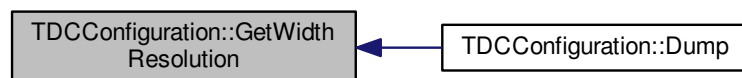


5.11.3.27 `uint8_t TDCConfiguration::GetVernierOffset () const [inline]`

Extract the offset in vernier decoding.

5.11.3.28 `WidthResolution TDCConfiguration::GetWidthResolution () const [inline]`

Here is the caller graph for this function:

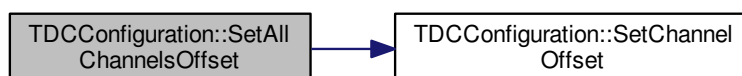


5.11.3.29 `word_t TDCConfiguration::GetWord (const unsigned int i) const [inline]`

Retrieve one subset from the setup word.

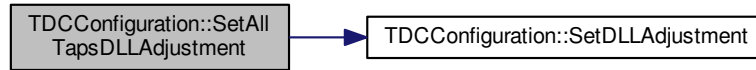
5.11.3.30 `void TDCConfiguration::SetAllChannelsOffset (uint16_t offset) [inline]`

Here is the call graph for this function:



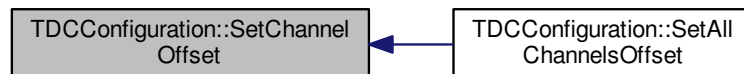
5.11.3.31 `void TDCConfiguration::SetAllTapsDLLAdjustment (uint8_t adj) [inline]`

Here is the call graph for this function:



5.11.3.32 `void TDCConfiguration::SetChannelOffset (int channel, uint16_t offset) [inline]`

Here is the caller graph for this function:



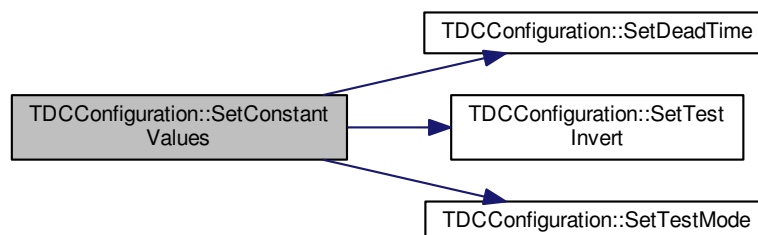
5.11.3.33 `void TDCConfiguration::SetCoarseCountOffset (uint16_t cco) [inline]`

Set offset for the coarse time counter.

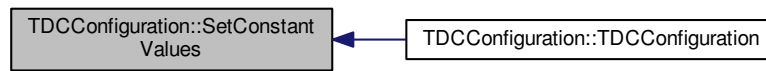
5.11.3.34 `void TDCConfiguration::SetConstantValues ()`

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

Here is the call graph for this function:



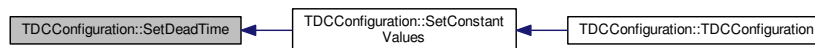
Here is the caller graph for this function:



5.11.3.35 `void TDCConfiguration::SetDeadTime (const DeadTime dt) [inline]`

Channel dead time between hits.

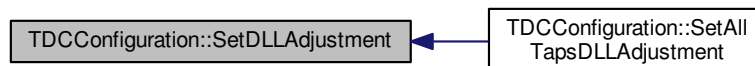
Here is the caller graph for this function:



5.11.3.36 `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:



5.11.3.37 `void TDCConfiguration::SetEdgeResolution (const EdgeResolution r) [inline]`

5.11.3.38 `void TDCConfiguration::SetEdgesPairing (const bool pair = true) [inline]`

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

5.11.3.39 `void TDCConfiguration::SetEnableError (const uint16_t & err) [inline]`

Enable internal error types for generation of global error signals.

5.11.3.40 `void TDCConfiguration::SetEnableErrorBypass (bool eb) [inline]`

Bypass TDC chip if global error signal is set.

5.11.3.41 `void TDCConfiguration::SetEnableErrorMark (bool em) [inline]`

Mark events with error if global error signal is set.

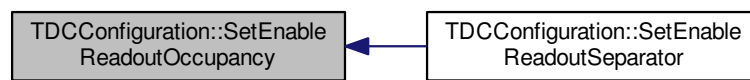
5.11.3.42 `void TDCConfiguration::SetEnableJTAGReadout (bool jr) [inline]`

Enable of read-out via JTAG.

5.11.3.43 `void TDCConfiguration::SetEnableReadoutOccupancy (const bool ro = true) [inline]`

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

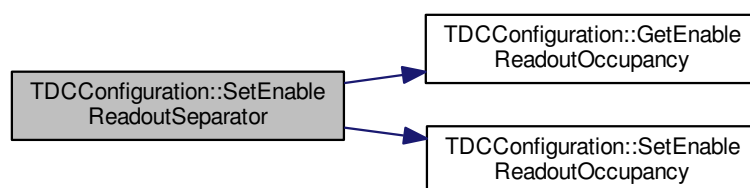
Here is the caller graph for this function:



5.11.3.44 `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:



5.11.3.45 `void TDCConfiguration::SetEnableSerial (bool es) [inline]`

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

5.11.3.46 `void TDCConfiguration::SetLeadingMode (const bool lead = true) [inline]`

Enable the detection of leading edges.

5.11.3.47 `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 unlimited.

5.11.3.48 `void TDCConfiguration::SetRCAdjustment (int tap, uint8_t adj) [inline]`

5.11.3.49 `void TDCConfiguration::SetRejectFIFOFull (bool rej = true) [inline]`

Reject hits when readout FIFO full.

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

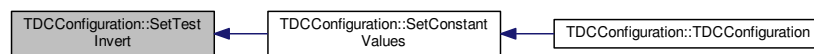
5.11.3.50 `void TDCConfiguration::SetSetupParity (const bool sp = true) [inline]`

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

5.11.3.51 `void TDCConfiguration::SetTestInvert (const bool ti = true) [inline]`

Automatic inversion of test pattern. Only used during production testing.

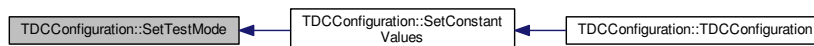
Here is the caller graph for this function:



5.11.3.52 `void TDCConfiguration::SetTestMode (const bool tm = true) [inline]`

Test mode where hit data are taken from coretest. Only used during production testing.

Here is the caller graph for this function:



5.11.3.53 `void TDCConfiguration::SetTrailingMode (const bool trail = true) [inline]`

Enable/disable the detection of trailing edges.

5.11.3.54 `void TDCConfiguration::SetTriggerCountOffset (uint16_t tco) [inline]`

Set offset for the trigger time tag counter.

5.11.3.55 void TDCCConfiguration::SetTriggerMatchingMode (const bool *trig* = true) [inline]

Set the enable status of trigger matching mode.

5.11.3.56 void TDCCConfiguration::SetVernierOffset (const uint8_t *vo*) [inline]

Set the offset in vernier decoding.

5.11.3.57 void TDCCConfiguration::SetWidthResolution (const WidthResolution *r*) [inline]

5.11.3.58 void TDCCConfiguration::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 files:

- include/TDCCConfiguration.h
- src/TDCCConfiguration.cpp

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:



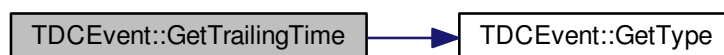
5.12.3.5 `unsigned int TDCEvent::GetTDCId () const [inline]`

Programmed identifier of master TDC.

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

Trailing edge measurement in programmed time resolution.

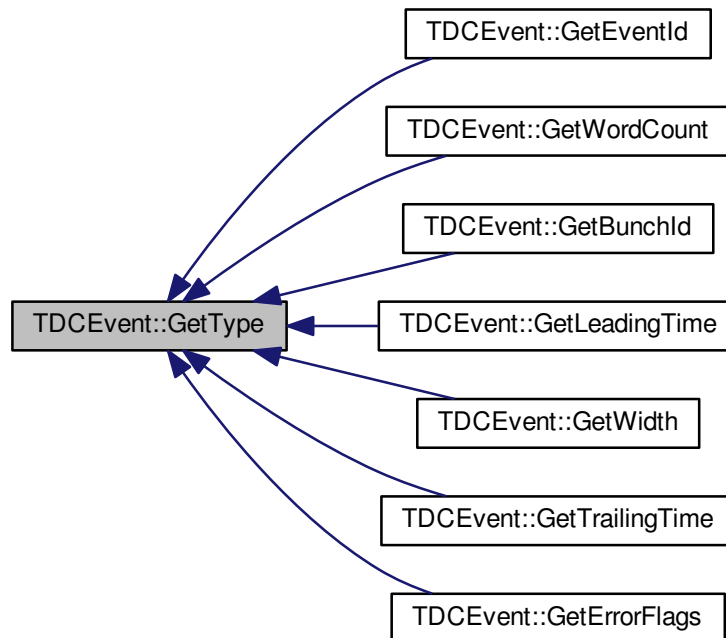
Here is the call graph for this function:



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.

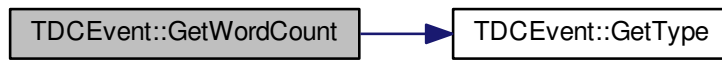
Here is the call graph for this function:



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:

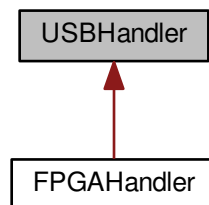
- `include/TDCEvent.h`

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

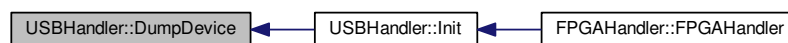
5.13.2.1 `USBHandler::USBHandler (const char * dev)`

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)`

Here is the caller graph for this function:



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 ()`

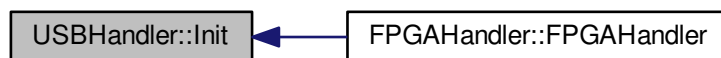
Pointer to a pointer of devices used to retrieve a list of them

A libusb session

Here is the call graph for this function:



Here is the caller graph for this function:



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 files:

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

Index

- ~Client
 - Client, [15](#)
- ~Exception
 - Exception, [19](#)
- ~FPGAHandler
 - FPGAHandler, [25](#)
- ~Message
 - Message, [31](#)
- ~Messenger
 - Messenger, [34](#)
- ~Socket
 - Socket, [40](#)
- ~SocketMessage
 - SocketMessage, [51](#)
- ~TDCCConfiguration
 - TDCCConfiguration, [57](#)
- ~TDCEvent
 - TDCEvent, [70](#)
- ~USBHandler
 - USBHandler, [74](#)
- AcceptConnections
 - Socket, [40](#)
- Bind
 - Socket, [41](#)
- Broadcast
 - Messenger, [34](#)
- CLIENT
 - Socket communication objects, [7](#)
- ChannelSelectError
 - HPTDC chip control, [11](#)
- Client, [13](#)
 - ~Client, [15](#)
 - Client, [15](#)
 - Connect, [15](#)
 - Disconnect, [15](#)
 - GetType, [16](#)
 - ParseMessage, [16](#)
 - Receive, [17](#)
 - Send, [17](#)
- CloseFile
 - FPGAHandler, [26](#)
- CoarseError
 - HPTDC chip control, [11](#)
- config
 - file_header_t, [23](#)
- Connect
 - Client, [15](#)
 - Messenger, [35](#)
- ControlParityError
 - HPTDC chip control, [11](#)
- Core_aux_clock
 - HPTDC chip control, [10](#)
- Core_clock_40
 - HPTDC chip control, [10](#)
- Core_pll_clock_160
 - HPTDC chip control, [10](#)
- Core_pll_clock_80
 - HPTDC chip control, [10](#)
- CoreClockSource
 - HPTDC chip control, [10](#)
- DETECTOR
 - Socket communication objects, [8](#)
- DLL_160MHz
 - HPTDC chip control, [10](#)
- DLL_320MHz
 - HPTDC chip control, [10](#)
- DLL_40MHz
 - HPTDC chip control, [10](#)
- DLL_Illegal
 - HPTDC chip control, [10](#)
- DLL_aux_clock
 - HPTDC chip control, [10](#)
- DLL_clock_40
 - HPTDC chip control, [10](#)
- DLL_pll_clock_160
 - HPTDC chip control, [10](#)
- DLL_pll_clock_320
 - HPTDC chip control, [10](#)
- DLL_pll_clock_40
 - HPTDC chip control, [10](#)
- DLLClockSource
 - HPTDC chip control, [10](#)
- DLLSpeedMode
 - HPTDC chip control, [10](#)
- DT_100ns
 - HPTDC chip control, [10](#)
- DT_10ns
 - HPTDC chip control, [10](#)
- DT_30ns
 - HPTDC chip control, [10](#)
- DT_5ns
 - HPTDC chip control, [10](#)
- DeadTime
 - HPTDC chip control, [10](#)
- Debug
 - HPTDC chip control, [11](#)

- Decode
 - HTTPMessage, 28
- Description
 - Exception, 19
- Disconnect
 - Client, 15
 - Messenger, 35
- Dump
 - Exception, 19
 - HTTPMessage, 29
 - Message, 31
 - SocketMessage, 51
 - TDCConfiguration, 57
- DumpConnected
 - Socket, 41
- DumpDevice
 - USBHandler, 74
- 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, 10
- EnabledError
 - HPTDC chip control, 10
- Encode
 - HTTPMessage, 29
- Error
 - HPTDC chip control, 11
- ErrorNumber
 - Exception, 20
- EventType
 - HPTDC chip control, 11
- Exception, 18
 - ~Exception, 19
 - Description, 19
 - Dump, 19
 - ErrorNumber, 20
 - Exception, 18
 - From, 21
 - Type, 21
 - TypeString, 21
- fBuffer
 - Socket, 47
- fMaster
 - Socket, 47
- FPGAHandler, 24
 - ~FPGAHandler, 25
 - CloseFile, 26
 - FPGAHandler, 25
 - GetConfiguration, 26
 - GetFilename, 26
 - GetType, 26
 - OpenFile, 26
 - ReadBuffer, 26
 - SetConfiguration, 26
- fPort
 - Socket, 47
- fReadFds
 - Socket, 47
- fSocketsConnected
 - Socket, 47
- fString
 - Message, 32
- Fetch
 - USBHandler, 74
- FetchMessage
 - Socket, 41
- file_header_t, 22
 - config, 23
 - magic, 23
 - run_id, 23
 - spill_id, 23
- From
 - Exception, 21
- GetBunchId
 - TDCEvent, 70
- GetChannelOffset
 - TDCConfiguration, 58
- GetCoarseCountOffset
 - TDCConfiguration, 59
- GetConfiguration
 - FPGAHandler, 26
- GetDLLAdjustment
 - TDCConfiguration, 59
- GetDeadTime
 - TDCConfiguration, 59
- GetEdgeResolution
 - TDCConfiguration, 60
- GetEdgesPairing
 - TDCConfiguration, 60
- GetEnableError
 - TDCConfiguration, 60
- GetEnableErrorBypass
 - TDCConfiguration, 61
- GetEnableErrorMark
 - TDCConfiguration, 61
- GetEnableJTAGReadout
 - TDCConfiguration, 61
- GetEnableReadoutOccupancy
 - TDCConfiguration, 61
- GetEnableReadoutSeparator
 - TDCConfiguration, 61

- GetEnableSerial
 - TDCConfiguration, [61](#)
- GetErrorFlags
 - TDCEvent, [70](#)
- GetEventId
 - TDCEvent, [70](#)
- GetFilename
 - FPGAHandler, [26](#)
- GetIntValue
 - SocketMessage, [51](#)
- GetKey
 - HTTPMessage, [29](#)
 - Message, [31](#)
 - SocketMessage, [51](#)
- GetLeadingMode
 - TDCConfiguration, [61](#)
- GetLeadingTime
 - TDCEvent, [71](#)
- GetMaxEventSize
 - TDCConfiguration, [61](#)
- GetNumWords
 - TDCConfiguration, [62](#)
- GetPort
 - Socket, [42](#)
- GetRCAdjustment
 - TDCConfiguration, [62](#)
- GetRejectFIFOFull
 - TDCConfiguration, [62](#)
- GetSetupParity
 - TDCConfiguration, [62](#)
- GetSocketId
 - Socket, [42](#)
- GetSocketType
 - Socket, [42](#)
- GetString
 - Message, [31](#)
 - SocketMessage, [52](#)
- GetTDCId
 - TDCEvent, [71](#)
- GetTestInvert
 - TDCConfiguration, [62](#)
- GetTestMode
 - TDCConfiguration, [62](#)
- GetTrailingMode
 - TDCConfiguration, [62](#)
- GetTrailingTime
 - TDCEvent, [71](#)
- GetTriggerCountOffset
 - TDCConfiguration, [63](#)
- GetTriggerLatency
 - TDCConfiguration, [63](#)
- GetTriggerMatchingMode
 - TDCConfiguration, [63](#)
- GetType
 - Client, [16](#)
 - FPGAHandler, [26](#)
 - Messenger, [36](#)
 - TDCEvent, [71](#)
- GetValue
 - SocketMessage, [52](#)
- GetVectorValue
 - SocketMessage, [52](#)
- GetVernierOffset
 - TDCConfiguration, [64](#)
- GetWidth
 - TDCEvent, [72](#)
- GetWidthResolution
 - TDCConfiguration, [64](#)
- GetWord
 - TDCConfiguration, [64](#)
- GetWordCount
 - TDCEvent, [72](#)
- GroupHeader
 - HPTDC chip control, [11](#)
- GroupTrailer
 - HPTDC chip control, [11](#)
- HPTDC chip control, [9](#)
 - ChannelSelectError, [11](#)
 - CoarseError, [11](#)
 - ControlParityError, [11](#)
 - Core_aux_clock, [10](#)
 - Core_clock_40, [10](#)
 - Core_pll_clock_160, [10](#)
 - Core_pll_clock_80, [10](#)
 - CoreClockSource, [10](#)
 - DLL_160MHz, [10](#)
 - DLL_320MHz, [10](#)
 - DLL_40MHz, [10](#)
 - DLL_Illegal, [10](#)
 - DLL_aux_clock, [10](#)
 - DLL_clock_40, [10](#)
 - DLL_pll_clock_160, [10](#)
 - DLL_pll_clock_320, [10](#)
 - DLL_pll_clock_40, [10](#)
 - DLLClockSource, [10](#)
 - DLLSpeedMode, [10](#)
 - DT_100ns, [10](#)
 - DT_10ns, [10](#)
 - DT_30ns, [10](#)
 - DT_5ns, [10](#)
 - DeadTime, [10](#)
 - Debug, [11](#)
 - 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, [10](#)
 - EnabledError, [10](#)
 - Error, [11](#)
 - EventType, [11](#)
 - GroupHeader, [11](#)
 - GroupTrailer, [11](#)

- IO_aux_clock, [11](#)
- IO_clock_40, [11](#)
- IO_pll_clock_160, [11](#)
- IO_pll_clock_80, [11](#)
- IOClockSource, [11](#)
- Invalid, [11](#)
- JTAGInstructionParityError, [11](#)
- L1BufferParityError, [11](#)
- LeadingEdge, [11](#)
- ReadoutFIFOParityError, [11](#)
- ReadoutStateError, [11](#)
- Serial_aux_clock, [11](#)
- Serial_pll_clock_160, [11](#)
- Serial_pll_clock_40, [11](#)
- Serial_pll_clock_80, [11](#)
- SerialClockSource, [11](#)
- SetupParityError, [11](#)
- TDCHeader, [11](#)
- TDCTrailer, [11](#)
- TrailingEdge, [11](#)
- TriggerFIFOParityError, [11](#)
- TriggerMatchingError, [11](#)
- VernierError, [11](#)
- W_100ns, [12](#)
- W_100ps, [12](#)
- W_12p5ns, [12](#)
- W_1p6ns, [12](#)
- W_200ns, [12](#)
- W_200ps, [12](#)
- W_25ns, [12](#)
- W_3p2ns, [12](#)
- W_400ns, [12](#)
- W_400ps, [12](#)
- W_50ns, [12](#)
- W_6p25ns, [12](#)
- W_800ns, [12](#)
- W_800ps, [12](#)
- WidthResolution, [11](#)
- HTTPMessage, [26](#)
 - Decode, [28](#)
 - Dump, [29](#)
 - Encode, [29](#)
 - GetKey, [29](#)
 - HTTPMessage, [28](#)
- INVALID
 - Socket communication objects, [7](#)
- IO_aux_clock
 - HPTDC chip control, [11](#)
- IO_clock_40
 - HPTDC chip control, [11](#)
- IO_pll_clock_160
 - HPTDC chip control, [11](#)
- IO_pll_clock_80
 - HPTDC chip control, [11](#)
- IOClockSource
 - HPTDC chip control, [11](#)
- Init
 - USBHandler, [74](#)
- Invalid
 - HPTDC chip control, [11](#)
- IsFromWeb
 - Message, [32](#)
- IsWebSocket
 - Socket, [43](#)
- JTAGInstructionParityError
 - HPTDC chip control, [11](#)
- L1BufferParityError
 - HPTDC chip control, [11](#)
- LeadingEdge
 - HPTDC chip control, [11](#)
- Listen
 - Socket, [43](#)
- ListenerInfo, [29](#)
 - name, [30](#)
 - type, [30](#)
- MASTER
 - Socket communication objects, [7](#)
- magic
 - file_header_t, [23](#)
- Message, [30](#)
 - ~Message, [31](#)
 - Dump, [31](#)
 - fString, [32](#)
 - GetKey, [31](#)
 - GetString, [31](#)
 - IsFromWeb, [32](#)
 - Message, [31](#)
- Messenger, [32](#)
 - ~Messenger, [34](#)
 - Broadcast, [34](#)
 - Connect, [35](#)
 - Disconnect, [35](#)
 - GetType, [36](#)
 - Messenger, [34](#)
 - Receive, [36](#)
 - Send, [36](#)
- name
 - ListenerInfo, [30](#)
- OpenFile
 - FPGAHandler, [26](#)
- ParseMessage
 - Client, [16](#)
- PrepareConnection
 - Socket, [44](#)
- ReadBuffer
 - FPGAHandler, [26](#)
- ReadoutFIFOParityError
 - HPTDC chip control, [11](#)
- ReadoutStateError
 - HPTDC chip control, [11](#)
- Receive

- Client, [17](#)
- Messenger, [36](#)
- run_id
 - file_header_t, [23](#)
- SelectConnections
 - Socket, [44](#)
- Send
 - Client, [17](#)
 - Messenger, [36](#)
- SendMessage
 - Socket, [44](#)
- Serial_aux_clock
 - HPTDC chip control, [11](#)
- Serial_pll_clock_160
 - HPTDC chip control, [11](#)
- Serial_pll_clock_40
 - HPTDC chip control, [11](#)
- Serial_pll_clock_80
 - HPTDC chip control, [11](#)
- SerialClockSource
 - HPTDC chip control, [11](#)
- SetAllChannelsOffset
 - TDCConfiguration, [64](#)
- SetAllTapsDLLAdjustment
 - TDCConfiguration, [64](#)
- SetChannelOffset
 - TDCConfiguration, [65](#)
- SetCoarseCountOffset
 - TDCConfiguration, [65](#)
- SetConfiguration
 - FPGAHandler, [26](#)
- SetConstantValues
 - TDCConfiguration, [65](#)
- SetDLLAdjustment
 - TDCConfiguration, [66](#)
- SetDeadTime
 - TDCConfiguration, [66](#)
- SetEdgeResolution
 - TDCConfiguration, [66](#)
- SetEdgesPairing
 - TDCConfiguration, [66](#)
- SetEnableError
 - TDCConfiguration, [66](#)
- SetEnableErrorBypass
 - TDCConfiguration, [66](#)
- SetEnableErrorMark
 - TDCConfiguration, [66](#)
- SetEnableJTAGReadout
 - TDCConfiguration, [67](#)
- SetEnableReadoutOccupancy
 - TDCConfiguration, [67](#)
- SetEnableReadoutSeparator
 - TDCConfiguration, [67](#)
- SetEnableSerial
 - TDCConfiguration, [67](#)
- SetKeyValue
 - SocketMessage, [53](#), [54](#)
- SetLeadingMode
 - TDCConfiguration, [67](#)
- SetMaxEventSize
 - TDCConfiguration, [67](#)
- SetPort
 - Socket, [45](#)
- SetRCAdjustment
 - TDCConfiguration, [68](#)
- SetRejectFIFOFull
 - TDCConfiguration, [68](#)
- SetSetupParity
 - TDCConfiguration, [68](#)
- SetSocketId
 - Socket, [45](#)
- SetTestInvert
 - TDCConfiguration, [68](#)
- SetTestMode
 - TDCConfiguration, [68](#)
- SetTrailingMode
 - TDCConfiguration, [68](#)
- SetTriggerCountOffset
 - TDCConfiguration, [68](#)
- SetTriggerMatchingMode
 - TDCConfiguration, [68](#)
- SetVernierOffset
 - TDCConfiguration, [69](#)
- SetWidthResolution
 - TDCConfiguration, [69](#)
- SetWord
 - TDCConfiguration, [69](#)
- SetupParityError
 - HPTDC chip control, [11](#)
- Socket, [38](#)
 - ~Socket, [40](#)
 - AcceptConnections, [40](#)
 - Bind, [41](#)
 - DumpConnected, [41](#)
 - fBuffer, [47](#)
 - fMaster, [47](#)
 - fPort, [47](#)
 - fReadFds, [47](#)
 - fSocketsConnected, [47](#)
 - FetchMessage, [41](#)
 - GetPort, [42](#)
 - GetSocketId, [42](#)
 - GetSocketType, [42](#)
 - IsWebSocket, [43](#)
 - Listen, [43](#)
 - PrepareConnection, [44](#)
 - SelectConnections, [44](#)
 - SendMessage, [44](#)
 - SetPort, [45](#)
 - SetSocketId, [45](#)
 - Socket, [40](#)
 - SocketCollection, [40](#)
 - Start, [45](#)
 - Stop, [46](#)
- Socket communication objects, [7](#)
 - CLIENT, [7](#)

- DETECTOR, 8
- INVALID, 7
- MASTER, 7
- SocketType, 7
- WEBSOCKET_CLIENT, 7
- SocketCollection
 - Socket, 40
- SocketMessage, 47
 - ~SocketMessage, 51
 - Dump, 51
 - GetIntValue, 51
 - GetKey, 51
 - GetString, 52
 - GetValue, 52
 - GetVectorValue, 52
 - SetKeyValue, 53, 54
 - SocketMessage, 49–51
- SocketType
 - Socket communication objects, 7
- spill_id
 - file_header_t, 23
- Start
 - Socket, 45
- Stop
 - Socket, 46
- TDCCConfiguration, 54
 - ~TDCCConfiguration, 57
 - Dump, 57
 - GetChannelOffset, 58
 - GetCoarseCountOffset, 59
 - GetDLLAdjustment, 59
 - GetDeadTime, 59
 - GetEdgeResolution, 60
 - GetEdgesPairing, 60
 - GetEnableError, 60
 - GetEnableErrorBypass, 61
 - GetEnableErrorMark, 61
 - GetEnableJTAGReadout, 61
 - GetEnableReadoutOccupancy, 61
 - GetEnableReadoutSeparator, 61
 - GetEnableSerial, 61
 - GetLeadingMode, 61
 - GetMaxEventSize, 61
 - GetNumWords, 62
 - GetRCAdjustment, 62
 - GetRejectFIFOFull, 62
 - GetSetupParity, 62
 - GetTestInvert, 62
 - GetTestMode, 62
 - GetTrailingMode, 62
 - GetTriggerCountOffset, 63
 - GetTriggerLatency, 63
 - GetTriggerMatchingMode, 63
 - GetVernierOffset, 64
 - GetWidthResolution, 64
 - GetWord, 64
 - SetAllChannelsOffset, 64
 - SetAllTapsDLLAdjustment, 64
 - SetChannelOffset, 65
 - SetCoarseCountOffset, 65
 - SetConstantValues, 65
 - SetDLLAdjustment, 66
 - SetDeadTime, 66
 - SetEdgeResolution, 66
 - SetEdgesPairing, 66
 - SetEnableError, 66
 - SetEnableErrorBypass, 66
 - SetEnableErrorMark, 66
 - SetEnableJTAGReadout, 67
 - SetEnableReadoutOccupancy, 67
 - SetEnableReadoutSeparator, 67
 - SetEnableSerial, 67
 - SetLeadingMode, 67
 - SetMaxEventSize, 67
 - SetRCAdjustment, 68
 - SetRejectFIFOFull, 68
 - SetSetupParity, 68
 - SetTestInvert, 68
 - SetTestMode, 68
 - SetTrailingMode, 68
 - SetTriggerCountOffset, 68
 - SetTriggerMatchingMode, 68
 - SetVernierOffset, 69
 - SetWidthResolution, 69
 - SetWord, 69
 - TDCCConfiguration, 57
- TDCEvent, 69
 - ~TDCEvent, 70
 - GetBunchId, 70
 - GetErrorFlags, 70
 - GetEventId, 70
 - GetLeadingTime, 71
 - GetTDCId, 71
 - GetTrailingTime, 71
 - GetType, 71
 - GetWidth, 72
 - GetWordCount, 72
 - TDCEvent, 70
- TDCHheader
 - HPTDC chip control, 11
- TDCTrailer
 - HPTDC chip control, 11
- TrailingEdge
 - HPTDC chip control, 11
- TriggerFIFOParityError
 - HPTDC chip control, 11
- TriggerMatchingError
 - HPTDC chip control, 11
- Type
 - Exception, 21
- type
 - ListenerInfo, 30
- TypeString
 - Exception, 21
- USBHandler, 73
 - ~USBHandler, 74

- DumpDevice, [74](#)
- Fetch, [74](#)
- Init, [74](#)
- USBHandler, [74](#)
- Write, [75](#)

VernierError

- HPTDC chip control, [11](#)

W_100ns

- HPTDC chip control, [12](#)

W_100ps

- HPTDC chip control, [12](#)

W_12p5ns

- HPTDC chip control, [12](#)

W_1p6ns

- HPTDC chip control, [12](#)

W_200ns

- HPTDC chip control, [12](#)

W_200ps

- HPTDC chip control, [12](#)

W_25ns

- HPTDC chip control, [12](#)

W_3p2ns

- HPTDC chip control, [12](#)

W_400ns

- HPTDC chip control, [12](#)

W_400ps

- HPTDC chip control, [12](#)

W_50ns

- HPTDC chip control, [12](#)

W_6p25ns

- HPTDC chip control, [12](#)

W_800ns

- HPTDC chip control, [12](#)

W_800ps

- HPTDC chip control, [12](#)

WEBSOCKET_CLIENT

- Socket communication objects, [7](#)

WidthResolution

- HPTDC chip control, [11](#)

Write

- USBHandler, [75](#)