2015-2016 Test beam Run Control

Generated by Doxygen 1.8.10

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 5

Module Documentation

5.1 Socket communication objects

Data Structures

- class Client

  Base client object for the socket.
- class 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.

5.1.1 Detailed Description

5.2 FPGA board control

Data Structures

- class DAQ::FPGAHandler

  Driver for timing detectors' FPGA readout.
- struct DAQ::QuickUSBHandler::FIFOFlags
- struct DAQ::QuickUSBHandler::Version
- class DAQ::QuickUSBHandler

  Generic QuickUSB communication handler.

5.2.1 Detailed Description

5.3 HPTDC chip control

Data Structures

- class TDCErrorFlag

  Error flags handler.
- class TDCEvent

  HPTDC event parser.
- class DAQ::TDC

  HPTDC object.

Enumerations

- enum AcquisitionMode { CONT_STORAGE, TRIG_MATCH }

  TDC acquisition mode.
- enum DAQ::TDC::AcquisitionMode { DAQ::TDC::CONT_STORAGE, DAQ::TDC::TRIG_MATCH }

  TDC acquisition mode.

5.3.1 Detailed Description

5.3.2 Enumeration Type Documentation

5.3.2.1 enum **AcquisitionMode**

TDC acquisition mode.

Author

Laurent Forthomme `laurent.forthomme@cern.ch`

Enumerator

CONT_STORAGE

TRIG_MATCH

5.3.2.2 enum **DAQ::TDC::AcquisitionMode**

TDC acquisition mode.

Enumerator

CONT_STORAGE

TRIG_MATCH

Chapter 6

Namespace Documentation

6.1 DAQ Namespace Reference

Data Structures

- class FPGAHandler

  Driver for timing detectors' FPGA readout.
- class QuickUSBHandler

  Generic QuickUSB communication handler.
- class TDC

  HPTDC object.

Functions

- std::ostream & operator<< (std::ostream &os, const QuickUSBHandler::HWRevision &rev)
- std::ostream & operator<< (std::ostream &os, const QuickUSBHandler::USBSpeed &sp)
- std::ostream & operator<< (std::ostream &os, const QuickUSBHandler::FPGAType &t)
- std::ostream & operator<< (std::ostream &os, const QuickUSBHandler::FIFOFlags &ff)

6.1.1 Function Documentation

6.1.1.1 std::ostream& DAQ::operator<< (std::ostream & *os*, const QuickUSBHandler::HWRevision & *rev*)

6.1.1.2 std::ostream& DAQ::operator<< (std::ostream & *os*, const QuickUSBHandler::USBSpeed & *sp*)

6.1.1.3 std::ostream& DAQ::operator<< (std::ostream & *os*, const QuickUSBHandler::FPGAType & *t*)

6.1.1.4 std::ostream& DAQ::operator<< (std::ostream & *os*, const QuickUSBHandler::FIFOFlags & *ff*)

6.2 DQM Namespace Reference

Data Structures

- class DQMProcess

  Handler for a common DQM process to run on the socket.
- class GastofCanvas
- class PPSCanvas
- class QuarticCanvas

Chapter 7

Data Structure Documentation

7.1 OnlineDBHandler::BurstInfo Struct Reference

```
#include <OnlineDBHandler.h>
```

Data Fields

- unsigned int burst_id
- unsigned int time_start

7.1.1 Field Documentation

7.1.1.1 unsigned int OnlineDBHandler::BurstInfo::burst_id

7.1.1.2 unsigned int OnlineDBHandler::BurstInfo::time_start

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

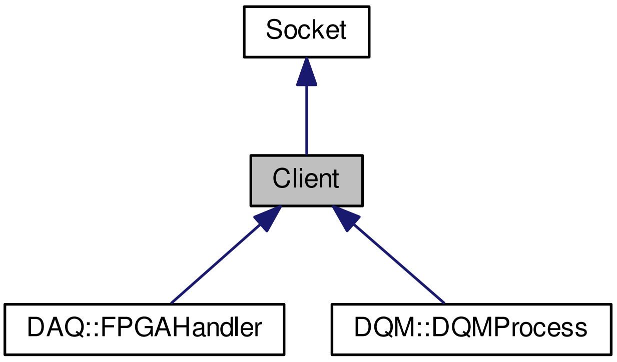
- include/OnlineDBHandler.h

7.2 Client Class Reference

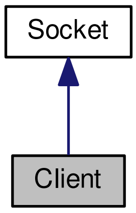
Base client object for the socket.

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

Inheritance diagram for Client:



Collaboration diagram for Client:



Public Member Functions

- Client ()

    General void client constructor.
- Client (int port)

    Bind a socket client to a given port.
- virtual ∼Client ()
- bool Connect (const SocketType &type=CLIENT)

    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 Send (const Exception &e) const
- SocketMessage SendAndReceive (const SocketMessage &m, const MessageKey &a) const
- void Receive ()

Receive a socket message from the master.

- SocketMessage Receive (const MessageKey &key)
- virtual void ParseMessage (const SocketMessage &m)

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

  Socket actor type retrieval method.

Private Member Functions

- void Announce ()

  Announce our entry on the socket to its master.

Private Attributes

- int fClientId
- bool fIsConnected
- SocketType fType

Additional Inherited Members

7.2.1 Detailed Description

Base client object for the socket.

Client object used by the server to send/receive commands from the messenger/broadcaster.

Author

Laurent Forthomme `laurent.forthomme@cern.ch`

Date

24 Mar 2015

7.2.2 Constructor & Destructor Documentation

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

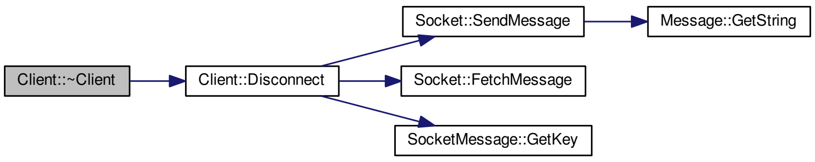
General void client constructor.

7.2.2.2 Client::Client (int *port*)

Bind a socket client to a given port.

7.2.2.3 Client::∼Client ( ) [virtual]

Here is the call graph for this function:

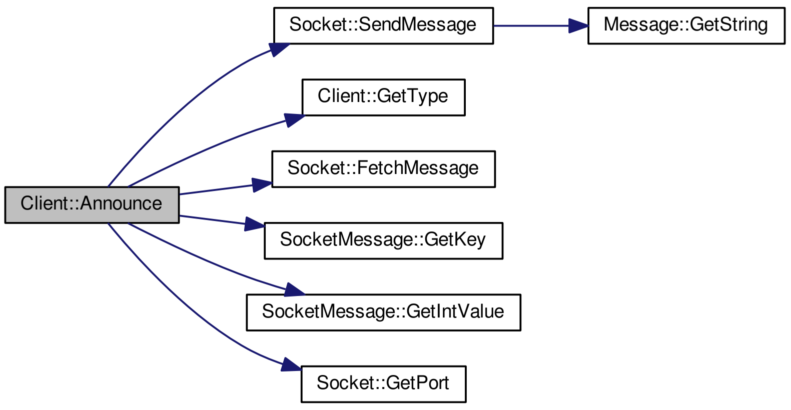


7.2.3 Member Function Documentation

7.2.3.1 void Client::Announce ( ) [private]

Announce our entry on the socket to its master.

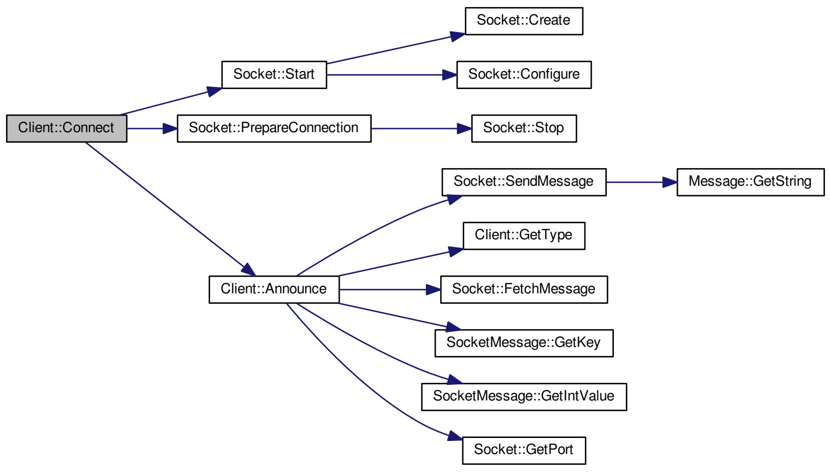
Here is the call graph for this function:



7.2.3.2 bool Client::Connect (const SocketType & *type* = CLIENT)

Bind this client to the socket.

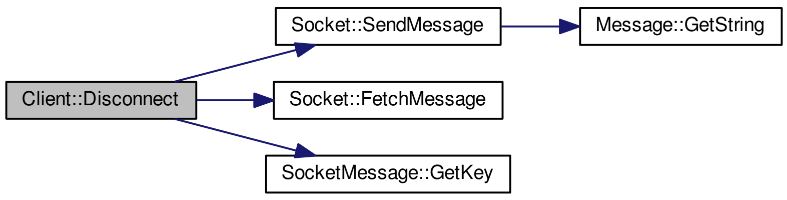
Here is the call graph for this function:



7.2.3.3 void Client::Disconnect ( )

Unbind this client from the socket.

Here is the call graph for this function:



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

Socket actor type retrieval method.

Reimplemented in DAQ::FPGAHandler.

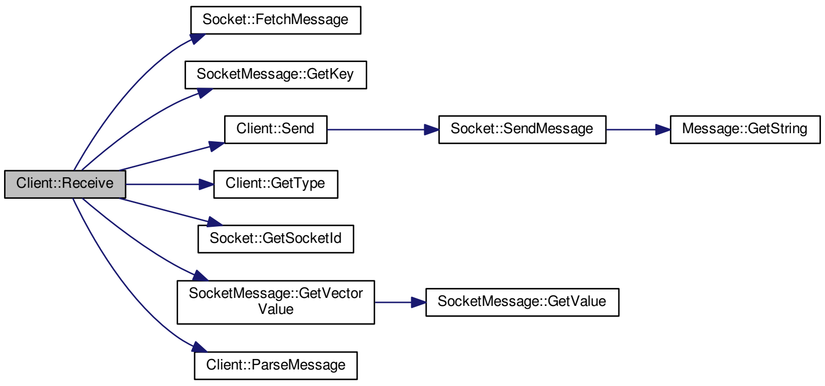
7.2.3.5 virtual void Client::ParseMessage (const SocketMessage & *m*) [inline], [virtual]

Parse a SocketMessage received from the master.

7.2.3.6 void Client::Receive ( )

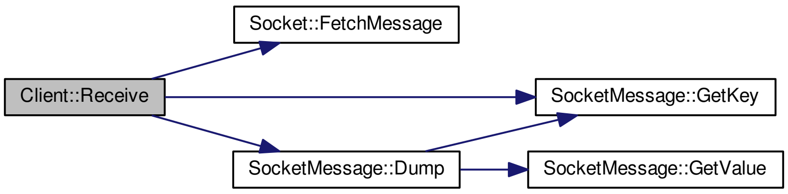
Receive a socket message from the master.

Here is the call graph for this function:



7.2.3.7 SocketMessage Client::Receive (const MessageKey & *key*)

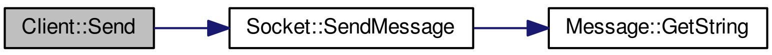
Here is the call graph for this function:



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

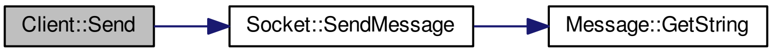
Send a message to the master through the socket.

Here is the call graph for this function:



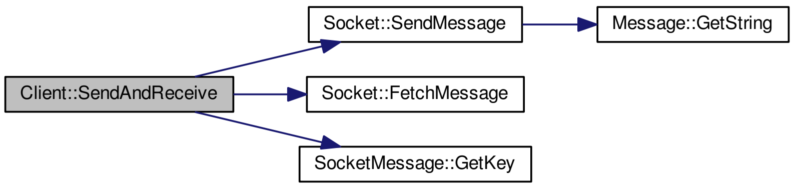
7.2.3.9 void Client::Send (const Exception & *e*) const [inline]

Here is the call graph for this function:



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

Here is the call graph for this function:



7.2.4 Field Documentation

7.2.4.1 int Client::fClientId [private]

7.2.4.2 bool Client::fIsConnected [private]

7.2.4.3 SocketType Client::fType [private]

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

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

7.3 DQM::GastofCanvas::Coord Struct Reference

Data Fields

- unsigned int x
- unsigned int y

7.3.1 Field Documentation

7.3.1.1 unsigned int DQM::GastofCanvas::Coord::x

7.3.1.2 unsigned int DQM::GastofCanvas::Coord::y

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

- include/GastofCanvas.h

7.4 DQM::QuarticCanvas::Coord Struct Reference

Data Fields

- unsigned int x
- unsigned int y

7.4.1 Field Documentation

7.4.1.1 unsigned int DQM::QuarticCanvas::Coord::x

7.4.1.2 unsigned int DQM::QuarticCanvas::Coord::y

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

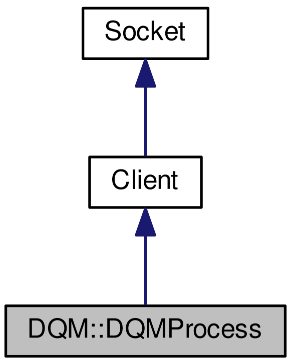
- include/QuarticCanvas.h

7.5 DQM::DQMProcess Class Reference

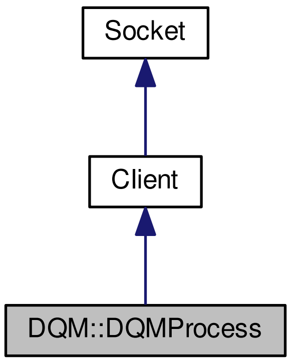
Handler for a common DQM process to run on the socket.

```
#include <DQMProcess.h>
```

Inheritance diagram for DQM::DQMProcess:



Collaboration diagram for DQM::DQMProcess:



Public Types

- enum Action { NewPlot = 0x0, UpdatedPlot = 0x1 }

Public Member Functions

- DQMProcess (int port, unsigned short order=0, const char ∗det_type="")
- ∼DQMProcess ()
- void Run (bool(∗fcn)(unsigned int addr, std::string filename, std::vector< std::string > ∗outputs), const Action &act=NewPlot)

  Run a DQM plotter making use of the board/output filename information.
- void Run (bool(∗fcn)(std::vector< std::string > ∗outputs), const Action &act=NewPlot)

  Run a DQM plotter without any information on the board/output filename.

Private Member Functions

- int ParseMessage (uint32_t ∗board_address, std::string ∗filename)
- bool IsInRun ()

Private Attributes

- unsigned short fOrder
- unsigned int fRunNumber
- std::string fDetectorType
- std::map< unsigned long, std::string > fAddressesCanProcess

Additional Inherited Members

7.5.1 Detailed Description

Handler for a common DQM process to run on the socket.

Author

Laurent Forthomme `laurent.forthomme@cern.ch`

Date

3 Aug 2015

7.5.2 Member Enumeration Documentation

7.5.2.1 enum DQM::DQMProcess::Action

Enumerator

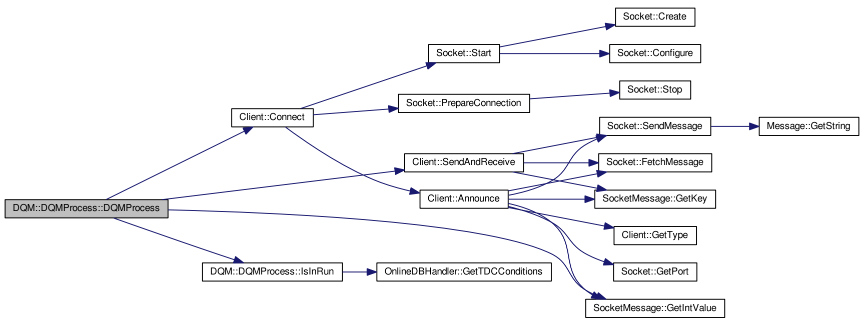
NewPlot

UpdatedPlot

7.5.3 Constructor & Destructor Documentation

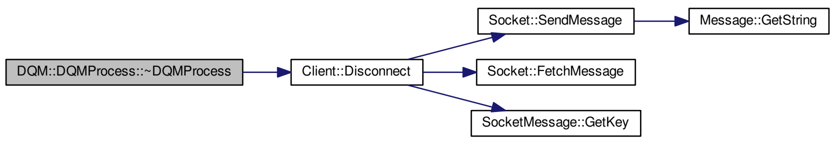
7.5.3.1 DQM::DQMProcess::DQMProcess (int *port*, unsigned short *order* = `0`, const char ∗ *det_type* = " ") `[inline]`

Here is the call graph for this function:



7.5.3.2 DQM::DQMProcess::∼DQMProcess ( ) `[inline]`

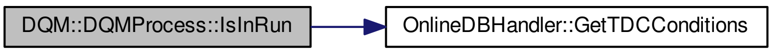
Here is the call graph for this function:



7.5.4 Member Function Documentation

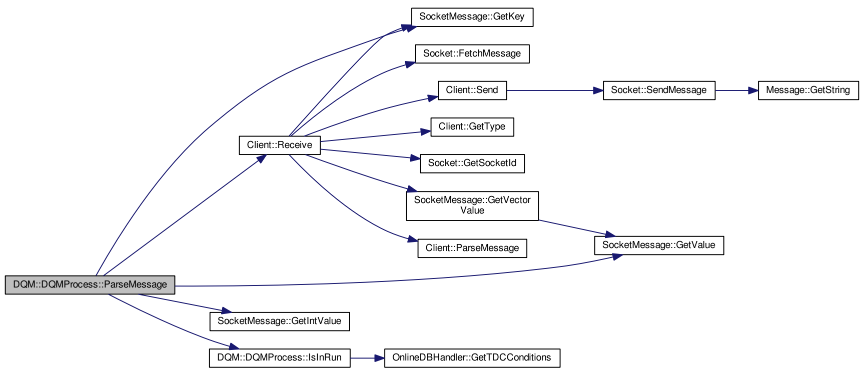
7.5.4.1 bool DQM::DQMProcess::IsInRun ( ) [inline], [private]

Here is the call graph for this function:



7.5.4.2 int DQM::DQMProcess::ParseMessage (uint32_t ∗ *board_address,* std::string ∗ *filename*) [inline], [private]

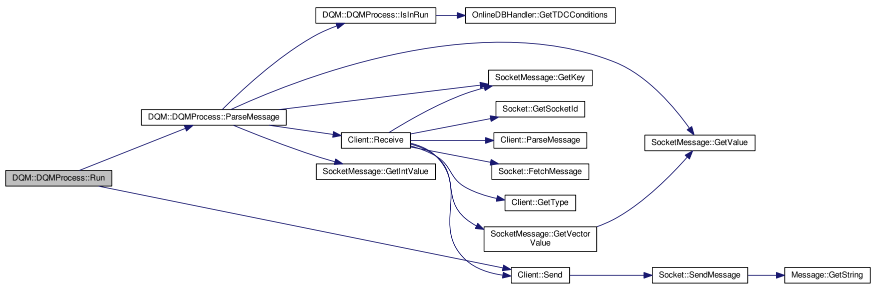
Here is the call graph for this function:



7.5.4.3 void DQM::DQMProcess::Run (bool(∗)(unsigned int addr, std::string filename, std::vector< std::string > ∗outputs) *fcn,* const Action & *act* = NewPlot) [inline]

Run a DQM plotter making use of the board/output filename information.

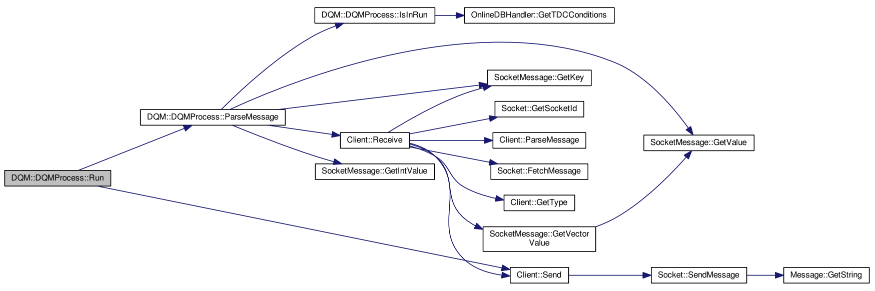
Here is the call graph for this function:



7.5.4.4 void DQM::DQMProcess::Run (bool(∗)(std::vector< std::string > ∗outputs) *fcn*, const Action & *act* = NewPlot) [inline]

Run a DQM plotter without any information on the board/output filename.

Here is the call graph for this function:



7.5.5 Field Documentation

7.5.5.1 std::map<unsigned long, std::string> DQM::DQMProcess::fAddressesCanProcess [private]

7.5.5.2 std::string DQM::DQMProcess::fDetectorType [private]

7.5.5.3 unsigned short DQM::DQMProcess::fOrder [private]

7.5.5.4 unsigned int DQM::DQMProcess::fRunNumber [private]

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

- include/DQMProcess.h

7.6 DAQ::QuickUSBHandler::FIFOFlags Struct Reference

```
#include <QuickUSBHandler.h>
```

Data Fields

- bool WriteFIFOFull
- bool WriteFIFOEmpty
- bool RDY1
- bool ReadFIFOFull
- bool ReadFIFOEmpty
- bool RDY0

Friends

- std::ostream & operator<< (std::ostream &out, const FIFOFlags &ff)

7.6.1 Friends And Related Function Documentation

7.6.1.1 std::ostream& operator<< (std::ostream & *out*, const **FIFOFlags** & *ff*) [friend]

7.6.2 Field Documentation

7.6.2.1 bool DAQ::QuickUSBHandler::FIFOFlags::RDY0

7.6.2.2 bool DAQ::QuickUSBHandler::FIFOFlags::RDY1

7.6.2.3 bool DAQ::QuickUSBHandler::FIFOFlags::ReadFIFOEmpty

7.6.2.4 bool DAQ::QuickUSBHandler::FIFOFlags::ReadFIFOFull

7.6.2.5 bool DAQ::QuickUSBHandler::FIFOFlags::WriteFIFOEmpty

7.6.2.6 bool DAQ::QuickUSBHandler::FIFOFlags::WriteFIFOFull

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

- daq/include/QuickUSBHandler.h

7.7 file_header_t Struct Reference

Header to the output files.

`#include <FileConstants.h>`

Data Fields

- uint32_t magic
- uint32_t run_id
- uint32_t spill_id
- uint8_t num_hptdc
- AcquisitionMode acq_mode
- DetectionMode det_mode

7.7.1 Detailed Description

Header to the output files.

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

Author

Laurent Forthomme `laurent.forthomme@cern.ch`

Date

14 Apr 2015

7.7.2 Field Documentation

7.7.2.1 AcquisitionMode file_header_t::acq_mode

7.7.2.2 DetectionMode file_header_t::det_mode

7.7.2.3 uint32_t file_header_t::magic

7.7.2.4 uint8_t file_header_t::num_hptdc

7.7.2.5 uint32_t file_header_t::run_id

7.7.2.6 uint32_t file_header_t::spill_id

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

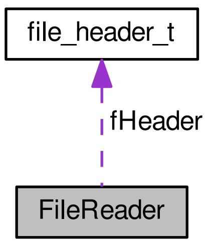
- include/FileConstants.h

7.8 FileReader Class Reference

Handler for a TDC output file readout.

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

Collaboration diagram for FileReader:



Public Member Functions

- FileReader ()
- FileReader (std::string name)

  Class constructor.
- ~FileReader ()
- void Open (std::string name)
- bool IsOpen () const
- void Clear ()
- void Dump () const
- unsigned int GetNumTDCs () const
- unsigned int GetRunId () const
- unsigned int GetBurstId () const

- unsigned int GetAcquisitionMode () const
- unsigned int GetDetectionMode () const
- unsigned long GetNumEvents () const
- bool GetNextEvent (TDCEvent ∗)
- bool GetNextMeasurement (unsigned int channel_id, TDCMeasurement ∗mc)

  Fetch the next full measurement on a given channel.

Private Attributes

- std::ifstream fFile
- file_header_t fHeader
- AcquisitionMode fReadoutMode
- time_t fWriteTime
- unsigned long fNumEvents

7.8.1 Detailed Description

Handler for a TDC output file readout.

Author

Laurent Forthomme `laurent.forthomme@cern.ch`

Date

Jun 2015

7.8.2 Constructor & Destructor Documentation

7.8.2.1 FileReader::FileReader ( ) `[inline]`

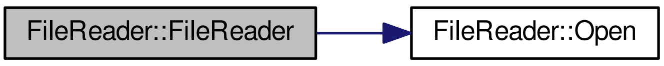
7.8.2.2 FileReader::FileReader (std::string *name*)

Class constructor.

Parameters

`in`	*name*	Path to the file to read
`in`	*ro*	Data readout mode (continuous storage or trigger matching)

Here is the call graph for this function:



7.8.2.3 FileReader::∼FileReader ( )

7.8.3 Member Function Documentation

7.8.3.1 void FileReader::Clear ( ) [inline]

7.8.3.2 void FileReader::Dump ( ) const

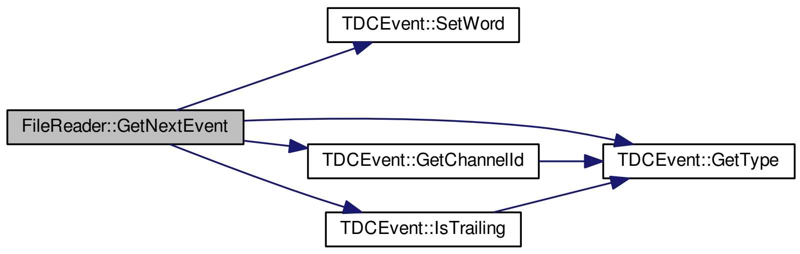
7.8.3.3 unsigned int FileReader::GetAcquisitionMode ( ) const [inline]

7.8.3.4 unsigned int FileReader::GetBurstId ( ) const [inline]

7.8.3.5 unsigned int FileReader::GetDetectionMode ( ) const [inline]

7.8.3.6 bool FileReader::GetNextEvent (TDCEvent ∗ *ev*)

Here is the call graph for this function:



7.8.3.7 bool FileReader::GetNextMeasurement (unsigned int *channel_id,* TDCMeasurement ∗ *mc*)

Fetch the next full measurement on a given channel.

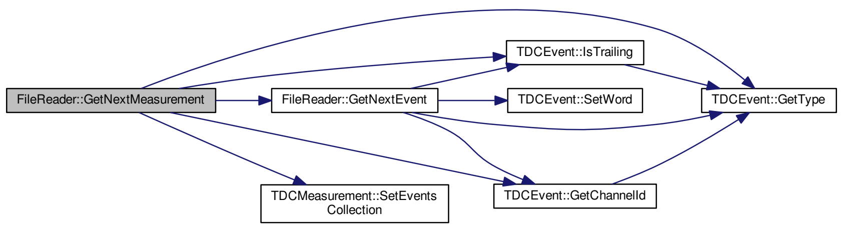
Parameters

in	*channel_id*	Unique identifier of the channel number to retrieve
out	*m*	A full measurement with leading, trailing times, ...

Returns

A boolean stating the success of retrieval operation

Here is the call graph for this function:



7.8.3.8 unsigned long FileReader::GetNumEvents ( ) const [inline]

7.8.3.9 unsigned int FileReader::GetNumTDCs ( ) const [inline]

7.8.3.10 unsigned int FileReader::GetRunId ( ) const [inline]

7.8.3.11 bool FileReader::IsOpen ( ) const [inline]

7.8.3.12 void FileReader::Open (std::string *name*)

7.8.4 Field Documentation

7.8.4.1 std::ifstream FileReader::fFile [private]

7.8.4.2 file_header_t FileReader::fHeader [private]

7.8.4.3 unsigned long FileReader::fNumEvents [private]

7.8.4.4 AcquisitionMode FileReader::fReadoutMode [private]

7.8.4.5 time_t FileReader::fWriteTime [private]

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

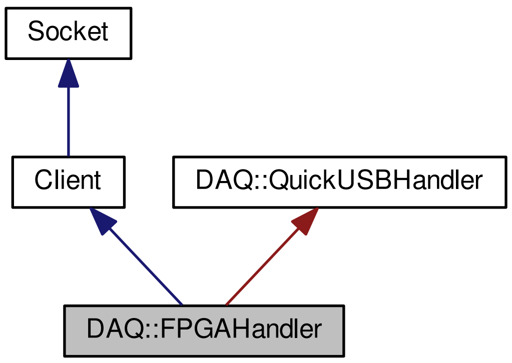
- include/FileReader.h
- src/FileReader.cpp

7.9 DAQ::FPGAHandler Class Reference

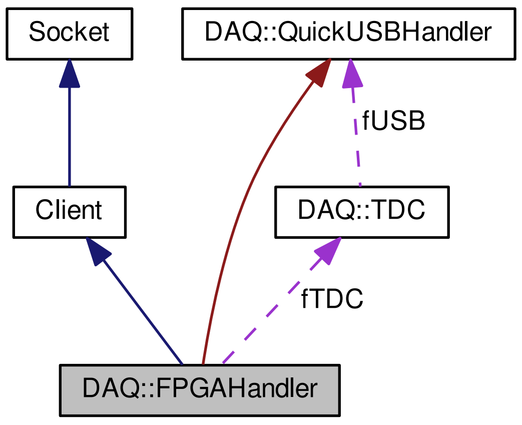
Driver for timing detectors' FPGA readout.

`#include <FPGAHandler.h>`

Inheritance diagram for DAQ::FPGAHandler:



Collaboration diagram for DAQ::FPGAHandler:



Public Member Functions

- FPGAHandler (int port, const char ∗dev)

    Bind to a FPGA through the USB protocol, and to the socket.
- ∼FPGAHandler ()
- void Stop ()
- 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.
- TDC ∗ GetTDC (unsigned int i=0)
- bool ErrorState ()
- void StartAcquisition ()
- void StopAcquisition ()
- SocketType GetType () const

    Socket actor type retrieval method.
- TDCControl GetTDCControl () const
- TDCStatus GetTDCStatus () const
- void SetTDCSetup (const TDCSetup &s)
- TDCSetup GetTDCSetup () const

Private Member Functions

- void RegisterTest () const
- void SendSetupWord () const
- void RetrieveSetupWord ()

Private Attributes

- std::string fFilename
- std::ofstream fOutput
- bool fIsFileOpen
- TDC $*$ fTDC [NUM_HPTDC]
- bool fIsTDCInReadout
- TDCSetup fSetupReg

Additional Inherited Members

7.9.1 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

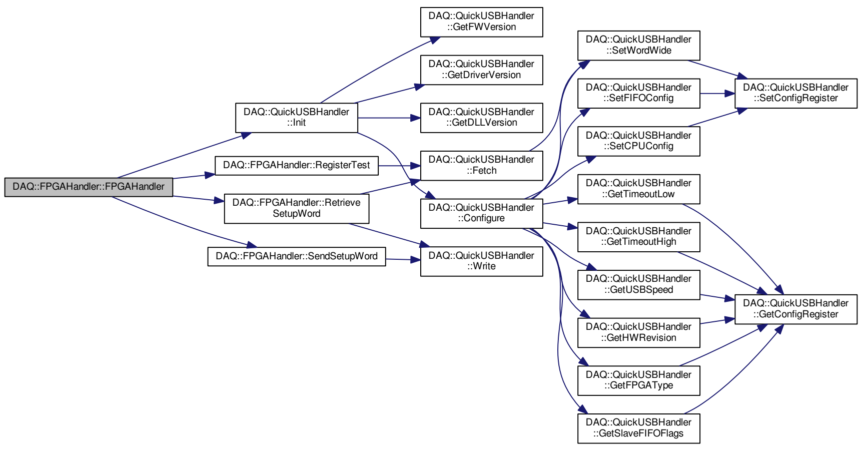
14 Apr 2015

7.9.2 Constructor & Destructor Documentation

7.9.2.1 DAQ::FPGAHandler::FPGAHandler (int *port*, const char $*$ *dev*)

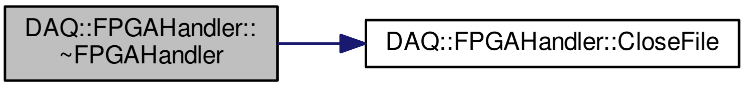
Bind to a FPGA through the USB protocol, and to the socket.

Here is the call graph for this function:


DAQ::QuickUSBHandler ::GetFWVersion
DAQ::QuickUSBHandler ::GetDriverVersion
DAQ::QuickUSBHandler ::Init
DAQ::QuickUSBHandler ::GetDLLVersion
DAQ::FPGAHandler::RegisterTest
DAQ::QuickUSBHandler ::Fetch
DAQ::FPGAHandler::FPGAHandler
DAQ::FPGAHandler::Retrieve SetupWord
DAQ::QuickUSBHandler ::Configure
DAQ::FPGAHandler::SendSetupWord
DAQ::QuickUSBHandler ::Write
DAQ::QuickUSBHandler ::SetWordWide
DAQ::QuickUSBHandler ::SetFIFOConfig
DAQ::QuickUSBHandler ::SetCPUConfig
DAQ::QuickUSBHandler ::GetTimeoutLow
DAQ::QuickUSBHandler ::GetTimeoutHigh
DAQ::QuickUSBHandler ::GetUSBSpeed
DAQ::QuickUSBHandler ::GetHWRevision
DAQ::QuickUSBHandler ::GetFPGAType
DAQ::QuickUSBHandler ::GetSlaveFIFOFlags
DAQ::QuickUSBHandler ::SetConfigRegister
DAQ::QuickUSBHandler ::GetConfigRegister

7.9.2.2 DAQ::FPGAHandler::∼FPGAHandler ( )

Here is the call graph for this function:



7.9.3 Member Function Documentation

7.9.3.1 void DAQ::FPGAHandler::CloseFile ( )

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

7.9.3.2 bool DAQ::FPGAHandler::ErrorState ( )

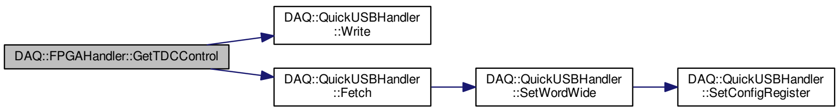
7.9.3.3 std::string DAQ::FPGAHandler::GetFilename ( ) const [inline]

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

7.9.3.4 TDC∗ DAQ::FPGAHandler::GetTDC (unsigned int *i* = 0) [inline]

7.9.3.5 TDCControl DAQ::FPGAHandler::GetTDCControl ( ) const

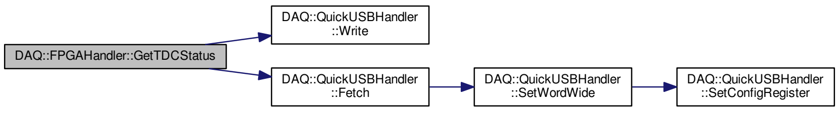
Here is the call graph for this function:


DAQ::FPGAHandler::GetTDCControl
DAQ::QuickUSBHandler ::Write
DAQ::QuickUSBHandler ::Fetch
DAQ::QuickUSBHandler ::SetWordWide
DAQ::QuickUSBHandler ::SetConfigRegister

7.9.3.6 TDCSetup DAQ::FPGAHandler::GetTDCSetup ( ) const [inline]

7.9.3.7 TDCStatus DAQ::FPGAHandler::GetTDCStatus ( ) const

Here is the call graph for this function:


DAQ::FPGAHandler::GetTDCStatus
DAQ::QuickUSBHandler ::Write
DAQ::QuickUSBHandler ::Fetch
DAQ::QuickUSBHandler ::SetWordWide
DAQ::QuickUSBHandler ::SetConfigRegister

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

Socket actor type retrieval method.

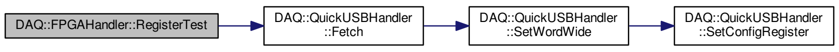
Reimplemented from Client.

7.9.3.9 void DAQ::FPGAHandler::OpenFile ( )

Open an output file to store header/HPTDC events.

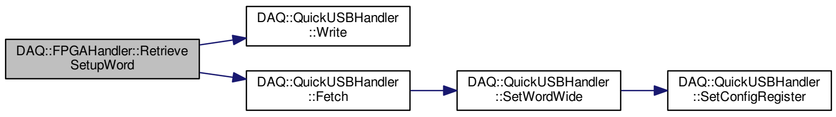
7.9.3.10 void DAQ::FPGAHandler::RegisterTest ( ) const [private]

Here is the call graph for this function:


DAQ::FPGAHandler::RegisterTest
DAQ::QuickUSBHandler ::Fetch
DAQ::QuickUSBHandler ::SetWordWide
DAQ::QuickUSBHandler ::SetConfigRegister

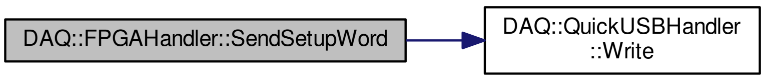
7.9.3.11 void DAQ::FPGAHandler::RetrieveSetupWord ( ) [private]

Here is the call graph for this function:


DAQ::FPGAHandler::Retrieve SetupWord
DAQ::QuickUSBHandler ::Write
DAQ::QuickUSBHandler ::Fetch
DAQ::QuickUSBHandler ::SetWordWide
DAQ::QuickUSBHandler ::SetConfigRegister

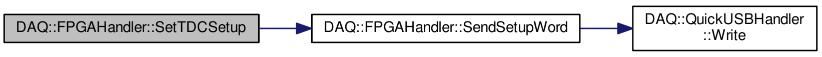
7.9.3.12 void DAQ::FPGAHandler::SendSetupWord ( ) const [private]

Here is the call graph for this function:



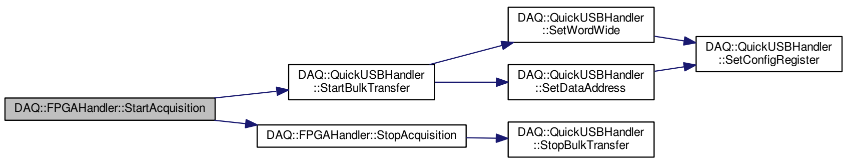
7.9.3.13 void DAQ::FPGAHandler::SetTDCSetup (const TDCSetup & *s*) [inline]

Here is the call graph for this function:


DAQ::FPGAHandler::SetTDCSetup
DAQ::FPGAHandler::SendSetupWord
DAQ::QuickUSBHandler ::Write

7.9.3.14 void DAQ::FPGAHandler::StartAcquisition ( )

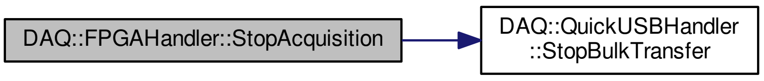
Here is the call graph for this function:


DAQ::FPGAHandler::StartAcquisition
DAQ::QuickUSBHandler ::StartBulkTransfer
DAQ::FPGAHandler::StopAcquisition
DAQ::QuickUSBHandler ::SetWordWide
DAQ::QuickUSBHandler ::SetDataAddress
DAQ::QuickUSBHandler ::StopBulkTransfer
DAQ::QuickUSBHandler ::SetConfigRegister

7.9.3.15 void DAQ::FPGAHandler::Stop ( ) [inline]

7.9.3.16 void DAQ::FPGAHandler::StopAcquisition ( )

Here is the call graph for this function:


DAQ::FPGAHandler::StopAcquisition
DAQ::QuickUSBHandler ::StopBulkTransfer

7.9.4 Field Documentation

7.9.4.1 std::string DAQ::FPGAHandler::fFilename [private]

7.9.4.2 bool DAQ::FPGAHandler::fIsFileOpen [private]

7.9.4.3 bool DAQ::FPGAHandler::fIsTDCInReadout [private]

7.9.4.4 std::ofstream DAQ::FPGAHandler::fOutput [private]

7.9.4.5 TDCSetup DAQ::FPGAHandler::fSetupReg [private]

7.9.4.6 TDC∗ DAQ::FPGAHandler::fTDC[NUM_HPTDC] `[private]`

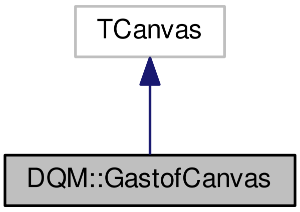
The documentation for this class was generated from the following files:

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

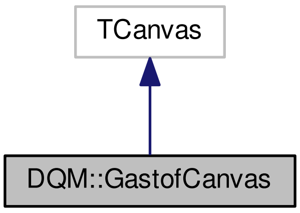
7.10 DQM::GastofCanvas Class Reference

`#include <GastofCanvas.h>`

Inheritance diagram for DQM::GastofCanvas:



Collaboration diagram for DQM::GastofCanvas:



Data Structures

- struct Coord

Public Member Functions

- GastofCanvas ()
- GastofCanvas (TString name, unsigned int width=500, unsigned int height=500, TString upper_label="")
- GastofCanvas (TString name, TString upper_label)
- virtual ∼GastofCanvas ()

- void SetRunInfo (unsigned int board_id, unsigned int run_id, unsigned int spill_id, TString date)
- void SetUpperLabel (TString text)
- void FillChannel (unsigned short nino_id, unsigned short channel_id, double content)
- TH2D ∗ Grid ()
- void Save (TString ext="png", TString path=".")

Private Member Functions

- void Build ()
- void DrawGrid ()
- Coord GetCoordinates (unsigned short nino_id, unsigned short channel_id) const

Private Attributes

- TPad ∗ c1
- TPad ∗ c2
- TH2D ∗ fHist
- double fWidth
- double fHeight
- TLegend ∗ fLegend
- double fLegendX
- double fLegendY
- unsigned int fLegendNumEntries
- TPaveText ∗ fLabel1
- TPaveText ∗ fLabel2
- TPaveText ∗ fLabel3
- TPaveText ∗ fLabel4
- TString fUpperLabelText
- TPaveText ∗ fUpperLabel
- bool fLabelsDrawn
- unsigned int fBoardId
- unsigned int fRunId
- unsigned int fSpillId
- TString fRunDate

7.10.1 Detailed Description

Author

Laurent Forthomme `laurent.forthomme@cern.ch`

Date

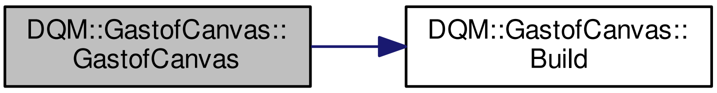
25 Jul 2015

7.10.2 Constructor & Destructor Documentation

7.10.2.1 DQM::GastofCanvas::GastofCanvas ( ) `[inline]`

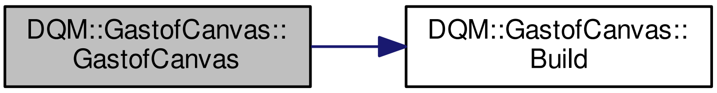
7.10.2.2 DQM::GastofCanvas::GastofCanvas (TString *name*, unsigned int *width* = `500`, unsigned int *height* = `500`, TString *upper_label* = `""`) `[inline]`

Here is the call graph for this function:



7.10.2.3 DQM::GastofCanvas::GastofCanvas (TString *name*, TString *upper_label*) `[inline]`

Here is the call graph for this function:



7.10.2.4 virtual DQM::GastofCanvas::∼GastofCanvas ( ) `[inline]`, `[virtual]`

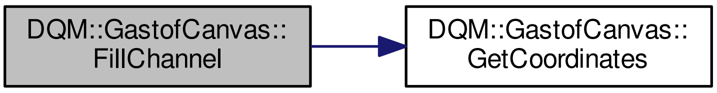
7.10.3 Member Function Documentation

7.10.3.1 void DQM::GastofCanvas::Build ( ) `[inline]`, `[private]`

7.10.3.2 void DQM::GastofCanvas::DrawGrid ( ) `[inline]`, `[private]`

7.10.3.3 void DQM::GastofCanvas::FillChannel (unsigned short *nino_id*, unsigned short *channel_id*, double *content*) `[inline]`

Here is the call graph for this function:

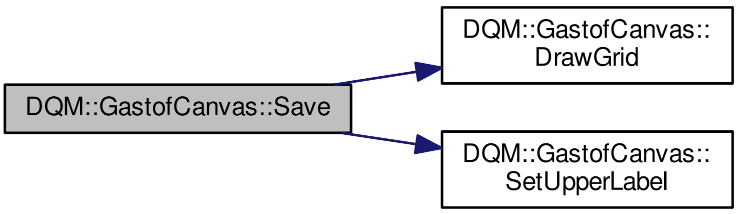


7.10.3.4 Coord DQM::GastofCanvas::GetCoordinates (unsigned short *nino_id,* unsigned short *channel_id*) const [inline], [private]

7.10.3.5 TH2D∗ DQM::GastofCanvas::Grid ( ) [inline]

7.10.3.6 void DQM::GastofCanvas::Save (TString *ext* = "png", TString *path* = ".") [inline]

Here is the call graph for this function:



7.10.3.7 void DQM::GastofCanvas::SetRunInfo (unsigned int *board_id,* unsigned int *run_id,* unsigned int *spill_id,* TString *date*) [inline]

7.10.3.8 void DQM::GastofCanvas::SetUpperLabel (TString *text*) [inline]

7.10.4 Field Documentation

7.10.4.1 TPad∗ DQM::GastofCanvas::c1 [private]

7.10.4.2 TPad ∗ DQM::GastofCanvas::c2 [private]

7.10.4.3 unsigned int DQM::GastofCanvas::fBoardId [private]

7.10.4.4 double DQM::GastofCanvas::fHeight [private]

7.10.4.5 TH2D∗ DQM::GastofCanvas::fHist [private]

7.10.4.6 TPaveText∗ DQM::GastofCanvas::fLabel1 [private]

7.10.4.7 TPaveText ∗ DQM::GastofCanvas::fLabel2 [private]

7.10.4.8 TPaveText ∗ DQM::GastofCanvas::fLabel3 [private]

7.10.4.9 TPaveText ∗ DQM::GastofCanvas::fLabel4 [private]

7.10.4.10 bool DQM::GastofCanvas::fLabelsDrawn [private]

7.10.4.11 TLegend∗ DQM::GastofCanvas::fLegend [private]

7.10.4.12 unsigned int DQM::GastofCanvas::fLegendNumEntries [private]

7.10.4.13 double DQM::GastofCanvas::fLegendX `[private]`

7.10.4.14 double DQM::GastofCanvas::fLegendY `[private]`

7.10.4.15 TString DQM::GastofCanvas::fRunDate `[private]`

7.10.4.16 unsigned int DQM::GastofCanvas::fRunId `[private]`

7.10.4.17 unsigned int DQM::GastofCanvas::fSpillId `[private]`

7.10.4.18 TPaveText∗ DQM::GastofCanvas::fUpperLabel `[private]`

7.10.4.19 TString DQM::GastofCanvas::fUpperLabelText `[private]`

7.10.4.20 double DQM::GastofCanvas::fWidth `[private]`

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

- include/GastofCanvas.h

7.11 Logger Class Reference

Redirect outputs to another output stream.

`#include <FileConstants.h>`

Public Member Functions

- Logger (std::ostream &lhs, std::ostream &rhs=std::cout)
- ∼Logger ()

Private Attributes

- std::ostream & fStream
- std::streambuf ∗const fBuffer

7.11.1 Detailed Description

Redirect outputs to another output stream.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 Logger::Logger (std::ostream & *lhs,* std::ostream & *rhs* = `std::cout`) `[inline]`

7.11.2.2 Logger::∼Logger ( ) `[inline]`

7.11.3 Field Documentation

7.11.3.1 std::streambuf∗ const Logger::fBuffer `[private]`

7.11.3.2 std::ostream& Logger::fStream `[private]`

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

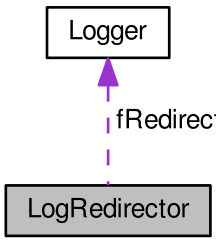
- include/FileConstants.h

7.12 LogRedirector Class Reference

Redirect output stream to a string.

`#include <FileConstants.h>`

Collaboration diagram for LogRedirector:



Public Member Functions

- LogRedirector (std::ostream &stm=std::cout)
- std::string contents () const

Private Attributes

- std::ostringstream fSS
- const Logger fRedirect

7.12.1 Detailed Description

Redirect output stream to a string.

Author

Laurent Forthomme `laurent.forthomme@cern.ch`

Date

3 Aug 2015

7.12.2 Constructor & Destructor Documentation

7.12.2.1 LogRedirector::LogRedirector (std::ostream & *stm* = `std::cout`) `[inline]`

7.12.3 Member Function Documentation

7.12.3.1 std::string LogRedirector::contents ( ) const [inline]

7.12.4 Field Documentation

7.12.4.1 const Logger LogRedirector::fRedirect [private]

7.12.4.2 std::ostringstream LogRedirector::fSS [private]

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

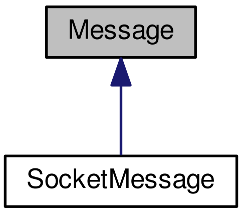
- include/FileConstants.h

7.13 Message Class Reference

Base socket message type.

`#include <Message.h>`

Inheritance diagram for Message:



Public Member Functions

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

Protected Attributes

- std::string fString

7.13.1 Detailed Description

Base socket message type.

Base handler for messages to be transmitted through the socket

Author

Laurent Forthomme `laurent.forthomme@cern.ch`

Date

6 Apr 2015

7.13.2 Constructor & Destructor Documentation

7.13.2.1 Message::Message ( ) `[inline]`

Void message constructor.

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

Construct a message from a string.

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

Construct a message from a string.

7.13.2.4 virtual Message::∼Message ( ) `[inline]`, `[virtual]`

7.13.3 Member Function Documentation

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

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

Placeholder for the MessageKey retrieval method.

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

Retrieve the string carried by this message as a whole.

7.13.3.4 bool Message::IsFromWeb ( ) const `[inline]`

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

7.13.4 Field Documentation

7.13.4.1 std::string Message::fString `[protected]`

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

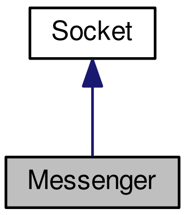
- include/Message.h

7.14 Messenger Class Reference

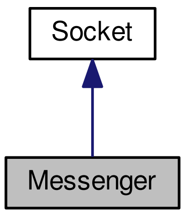
Base master object for the socket.

`#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 SendAll (const Socket::SocketType &type, const Message &m) const

  Send any type of message to all clients of one type.
- void SendAll (const Socket::SocketType &type, const Exception &e) const

- void Receive ()

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

  Emit a message to all clients connected through the socket.
- void StartAcquisition ()

  Start the data acquisition.
- void StopAcquisition ()
- SocketType GetType () const

  Socket actor type retrieval method.

Private Member Functions

- void AddClient ()

  Add a client to listen to.
- void DisconnectClient (int sid, MessageKey key, bool force=false)

  Disconnect a client.
- void SwitchClientType (int sid, Socket::SocketType type)
- void ProcessMessage (SocketMessage m, int sid)

  Process a message received from the socket.

Private Attributes

- int fNumAttempts
- pid_t fPID
- int fStdoutPipe [2]
- int fStderrPipe [2]

Additional Inherited Members

7.14.1 Detailed Description

Base master object for the socket.

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

Author

Laurent Forthomme `laurent.forthomme@cern.ch`

Date

23 Mar 2015

7.14.2 Constructor & Destructor Documentation

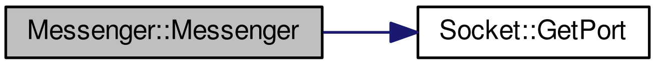
7.14.2.1 Messenger::Messenger ( )

Build a void master object or socket actor.

7.14.2.2 Messenger::Messenger (int *port*)

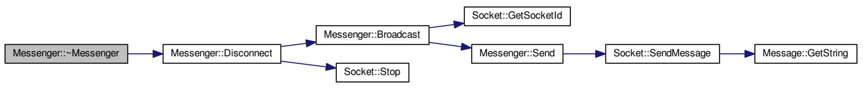
Build a master object to control the socket.

Here is the call graph for this function:



7.14.2.3 Messenger::∼Messenger ( )

Here is the call graph for this function:



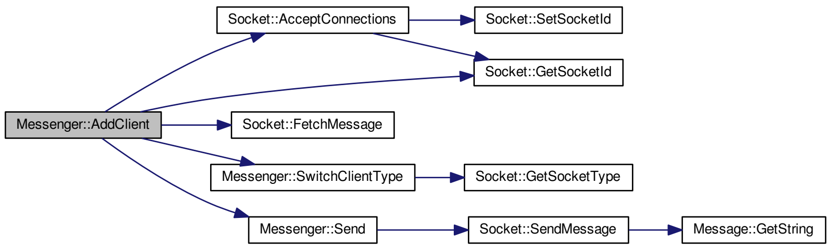
7.14.3 Member Function Documentation

7.14.3.1 void Messenger::AddClient ( ) `[private]`

Add a client to listen to.

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

Here is the call graph for this function:



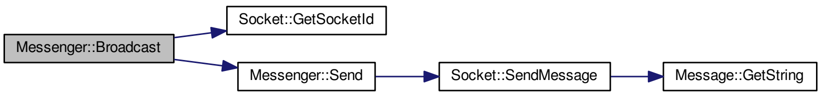
7.14.3.2 void Messenger::Broadcast (const Message & *m*) const

Emit a message to all clients connected through the socket.

Parameters

`in`	*m*	Message to transmit

Here is the call graph for this function:

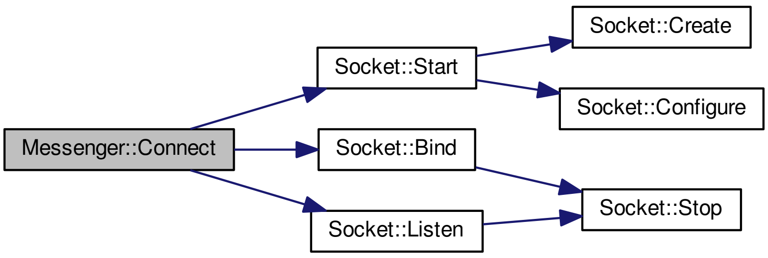


7.14.3.3 bool Messenger::Connect ( )

Connect the master to the socket.

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

Here is the call graph for this function:

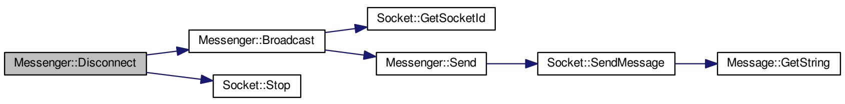


7.14.3.4 void Messenger::Disconnect ( )

Remove the master and destroy the socket.

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

Here is the call graph for this function:



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

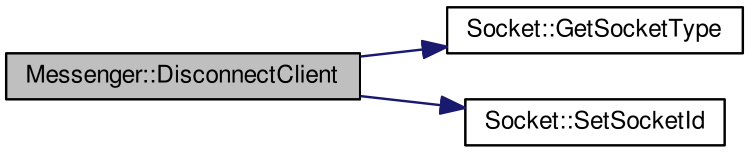
Disconnect a client.

Ask to a client to disconnect from this socket.

Parameters

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

Here is the call graph for this function:



7.14.3.6 **SocketType** Messenger::GetType ( ) const [inline]

Socket actor type retrieval method.

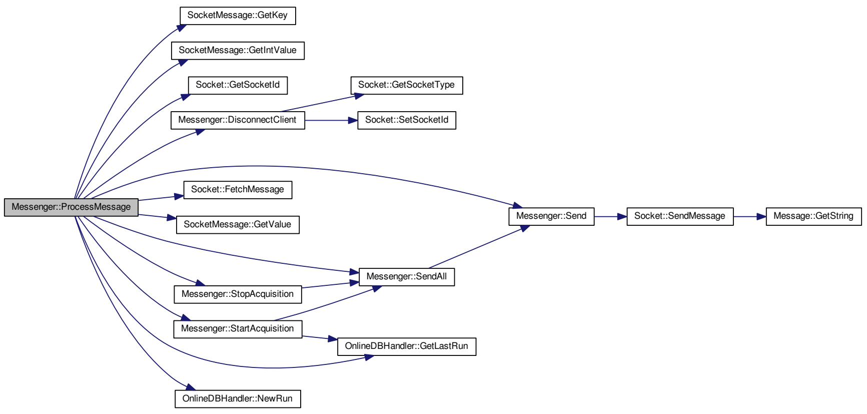
7.14.3.7 void Messenger::ProcessMessage (**SocketMessage** *m,* int *sid*) [private]

Process a message received from the socket.

Parameters

in	*Unique*	identifier of the client sending the message

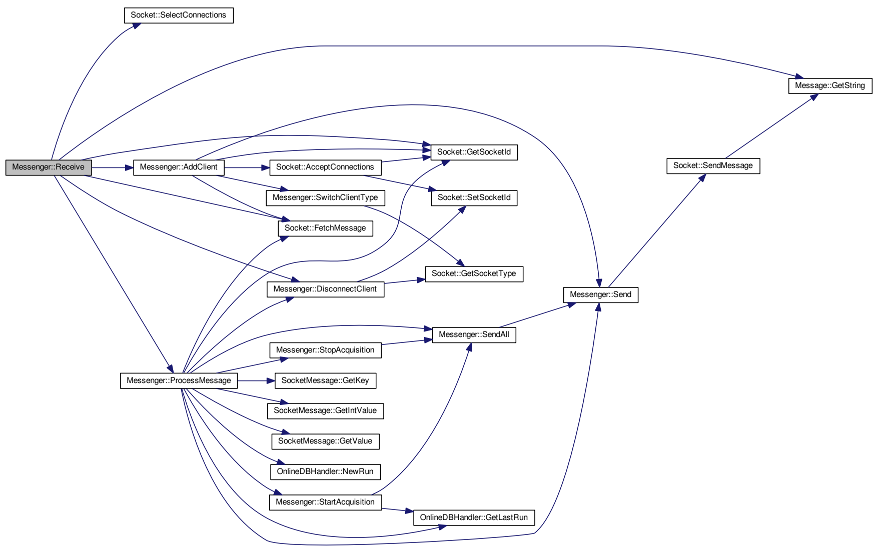
Here is the call graph for this function:



7.14.3.8 void Messenger::Receive ( )

Handle a message reception from a client.

Here is the call graph for this function:



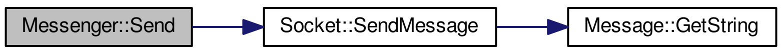
7.14.3.9 void Messenger::Send (const Message & *m*, int *sid*) const

Send any type of message to any client.

Parameters

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

Here is the call graph for this function:



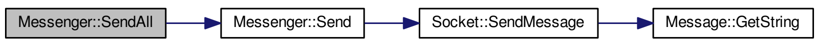
7.14.3.10 void Messenger::SendAll (const Socket::SocketType & *type*, const Message & *m*) const [inline]

Send any type of message to all clients of one type.

Parameters

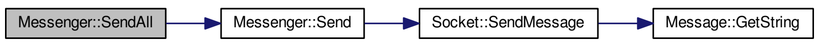
in	*type*	Client type
in	*m*	Message to transmit

Here is the call graph for this function:



7.14.3.11 void Messenger::SendAll (const Socket::SocketType & *type*, const Exception & *e*) const [inline]

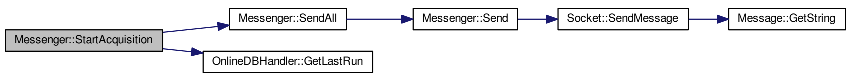
Here is the call graph for this function:



7.14.3.12 void Messenger::StartAcquisition ( )

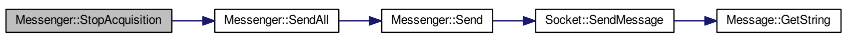
Start the data acquisition.

Here is the call graph for this function:



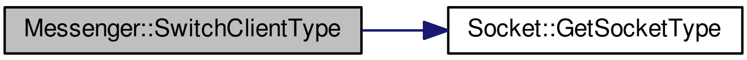
7.14.3.13 void Messenger::StopAcquisition ( )

Here is the call graph for this function:



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

Here is the call graph for this function:



7.14.4 Field Documentation

7.14.4.1 int Messenger::fNumAttempts `[private]`

7.14.4.2 pid_t Messenger::fPID `[private]`

7.14.4.3 int Messenger::fStderrPipe[2] `[private]`

7.14.4.4 int Messenger::fStdoutPipe[2] `[private]`

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

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

7.15 OnlineDBHandler Class Reference

Handler for the run information online database.

`#include <OnlineDBHandler.h>`

Data Structures

- struct BurstInfo
- struct TDCConditions

Public Types

- typedef std::map< unsigned int, unsigned int > RunCollection
- typedef std::vector< BurstInfo > BurstInfos
- typedef std::vector< TDCConditions > TDCConditionsCollection

Public Member Functions

- OnlineDBHandler (std::string path=std::string(std::getenv("PPS_PATH"))+"/run_infos.db")
- ∼OnlineDBHandler ()
- void NewRun ()
- void NewBurst ()
- RunCollection GetRuns () const

- unsigned int GetLastRun () const

  Retrieve the last run acquired.
- int GetLastBurst (unsigned int run) const
- BurstInfos GetRunInfo (unsigned int run) const

  Retrieve information on a given run (spill IDs / timestamp)
- void SetTDCConditions (unsigned short tdc_id, unsigned long tdc_address, unsigned short tdc_acq_mode, unsigned short tdc_det_mode, std::string detector)
- TDCConditionsCollection GetTDCConditions (unsigned int run_id) const
- void SetHVConditions (unsigned short channel_id, unsigned int vmax, unsigned imax)

Private Member Functions

- void BuildTables ()
- template<class T >
  std::vector< std::vector< T > > Select (std::string req, int num_fields=-1) const

Private Attributes

- sqlite3 ∗ fDB

7.15.1 Detailed Description

Handler for the run information online database.

Author

Laurent Forthomme `laurent.forthomme@cern.ch`

Date

3 Aug 2015

7.15.2 Member Typedef Documentation

7.15.2.1 typedef std::vector<BurstInfo> OnlineDBHandler::BurstInfos

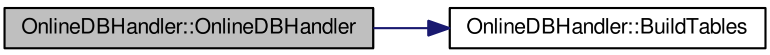
7.15.2.2 typedef std::map<unsigned int, unsigned int> OnlineDBHandler::RunCollection

7.15.2.3 typedef std::vector<TDCConditions> OnlineDBHandler::TDCConditionsCollection

7.15.3 Constructor & Destructor Documentation

7.15.3.1 OnlineDBHandler::OnlineDBHandler (std::string *path* = `std::string(std::getenv("PPS_↩ PATH"))+"/run_infos.db"`) `[inline]`

Here is the call graph for this function:


OnlineDBHandler::OnlineDBHandler
OnlineDBHandler::BuildTables

7.15.3.2 OnlineDBHandler::∼OnlineDBHandler ( ) [inline]

7.15.4 Member Function Documentation

7.15.4.1 void OnlineDBHandler::BuildTables ( ) [inline], [private]

7.15.4.2 int OnlineDBHandler::GetLastBurst (unsigned int *run*) const [inline]

7.15.4.3 unsigned int OnlineDBHandler::GetLastRun ( ) const [inline]

Retrieve the last run acquired.

7.15.4.4 BurstInfos OnlineDBHandler::GetRunInfo (unsigned int *run*) const [inline]

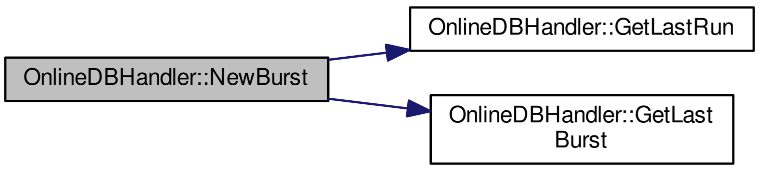
Retrieve information on a given run (spill IDs / timestamp)

7.15.4.5 RunCollection OnlineDBHandler::GetRuns ( ) const [inline]

7.15.4.6 TDCConditionsCollection OnlineDBHandler::GetTDCConditions (unsigned int *run_id*) const [inline]

7.15.4.7 void OnlineDBHandler::NewBurst ( ) [inline]

Here is the call graph for this function:

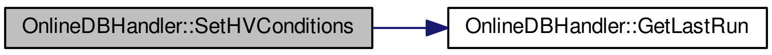

OnlineDBHandler::NewBurst
OnlineDBHandler::GetLastRun
OnlineDBHandler::GetLast Burst

7.15.4.8 void OnlineDBHandler::NewRun ( ) [inline]

7.15.4.9 template<class T > std::vector< std::vector<T> > OnlineDBHandler::Select (std::string *req*, int *num_fields* = −1) const [inline], [private]

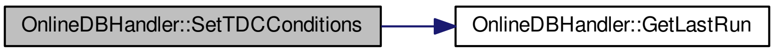
7.15.4.10 void OnlineDBHandler::SetHVConditions (unsigned short *channel_id*, unsigned int *vmax*, unsigned *imax*) [inline]

Here is the call graph for this function:



7.15.4.11 void OnlineDBHandler::SetTDCConditions (unsigned short *tdc_id*, unsigned long *tdc_address*, unsigned short *tdc_acq_mode*, unsigned short *tdc_det_mode*, std::string *detector*) `[inline]`

Here is the call graph for this function:


OnlineDBHandler::SetTDCConditions
OnlineDBHandler::GetLastRun

7.15.5 Field Documentation

7.15.5.1 sqlite3∗ OnlineDBHandler::fDB `[private]`

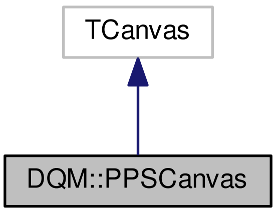
The documentation for this class was generated from the following file:

- include/OnlineDBHandler.h

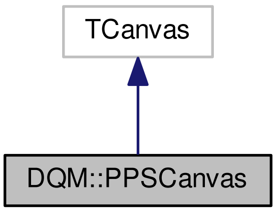
7.16 DQM::PPSCanvas Class Reference

`#include <PPSCanvas.h>`

Inheritance diagram for DQM::PPSCanvas:



Collaboration diagram for DQM::PPSCanvas:



Public Member Functions

- PPSCanvas ()
- PPSCanvas (TString name, unsigned int width=500, unsigned int height=500, TString upper_label="")
- PPSCanvas (TString name, TString upper_label)
- virtual ∼PPSCanvas ()
- void SetRunInfo (unsigned int run_id, TString date)
- void SetUpperLabel (TString text)
- TPad ∗ Grid ()
- void Save (TString ext="png", TString path=".")

Private Member Functions

- void Build ()
- void DrawGrid ()

Private Attributes

- TPad ∗ c1
- TPad ∗ c2
- double fWidth
- double fHeight
- TLegend ∗ fLegend
- double fLegendX
- double fLegendY
- unsigned int fLegendNumEntries
- TPaveText ∗ fLabel1
- TPaveText ∗ fLabel2
- TPaveText ∗ fLabel3
- TString fUpperLabelText
- TPaveText ∗ fUpperLabel
- bool fLabelsDrawn
- unsigned int fRunId
- TString fRunDate

7.16.1 Detailed Description

Author

Laurent Forthomme `laurent.forthomme@cern.ch`

Date

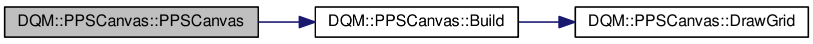
3 Aug 2015

7.16.2 Constructor & Destructor Documentation

7.16.2.1 DQM::PPSCanvas::PPSCanvas ( ) `[inline]`

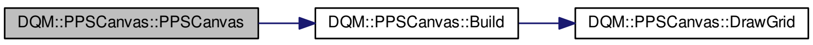
7.16.2.2 DQM::PPSCanvas::PPSCanvas (TString *name*, unsigned int *width* = `500`, unsigned int *height* = `500`, TString *upper_label* = " ") `[inline]`

Here is the call graph for this function:



7.16.2.3 DQM::PPSCanvas::PPSCanvas (TString *name*, TString *upper_label*) `[inline]`

Here is the call graph for this function:

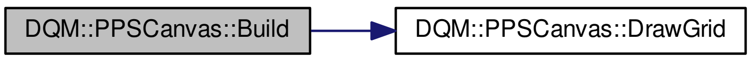


7.16.2.4 virtual DQM::PPSCanvas::∼PPSCanvas ( ) `[inline]`, `[virtual]`

7.16.3 Member Function Documentation

7.16.3.1 void DQM::PPSCanvas::Build ( ) `[inline]`, `[private]`

Here is the call graph for this function:

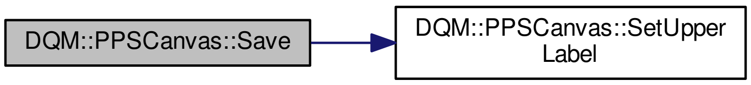


7.16.3.2 void DQM::PPSCanvas::DrawGrid ( ) [inline], [private]

7.16.3.3 TPad∗ DQM::PPSCanvas::Grid ( ) [inline]

7.16.3.4 void DQM::PPSCanvas::Save (TString *ext* = "png", TString *path* = ".") [inline]

Here is the call graph for this function:



7.16.3.5 void DQM::PPSCanvas::SetRunInfo (unsigned int *run_id*, TString *date*) [inline]

7.16.3.6 void DQM::PPSCanvas::SetUpperLabel (TString *text*) [inline]

7.16.4 Field Documentation

7.16.4.1 TPad∗ DQM::PPSCanvas::c1 [private]

7.16.4.2 TPad ∗ DQM::PPSCanvas::c2 [private]

7.16.4.3 double DQM::PPSCanvas::fHeight [private]

7.16.4.4 TPaveText∗ DQM::PPSCanvas::fLabel1 [private]

7.16.4.5 TPaveText ∗ DQM::PPSCanvas::fLabel2 [private]

7.16.4.6 TPaveText ∗ DQM::PPSCanvas::fLabel3 [private]

7.16.4.7 bool DQM::PPSCanvas::fLabelsDrawn [private]

7.16.4.8 TLegend∗ DQM::PPSCanvas::fLegend [private]

7.16.4.9 unsigned int DQM::PPSCanvas::fLegendNumEntries [private]

7.16.4.10 double DQM::PPSCanvas::fLegendX [private]

7.16.4.11 double DQM::PPSCanvas::fLegendY [private]

7.16.4.12 TString DQM::PPSCanvas::fRunDate [private]

7.16.4.13 unsigned int DQM::PPSCanvas::fRunId [private]

7.16.4.14 TPaveText∗ DQM::PPSCanvas::fUpperLabel [private]

7.16.4.15 TString DQM::PPSCanvas::fUpperLabelText [private]

7.16.4.16 double DQM::PPSCanvas::fWidth `[private]`

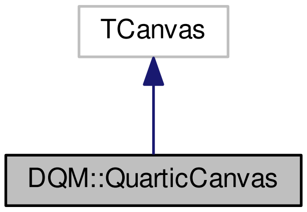
The documentation for this class was generated from the following file:

- include/PPSCanvas.h

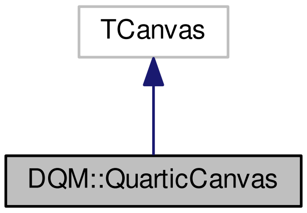
7.17 DQM::QuarticCanvas Class Reference

`#include <QuarticCanvas.h>`

Inheritance diagram for DQM::QuarticCanvas:



Collaboration diagram for DQM::QuarticCanvas:



Data Structures

- struct Coord

Public Member Functions

- QuarticCanvas ()
- QuarticCanvas (TString name, unsigned int width=500, unsigned int height=500, TString upper_label="")
- QuarticCanvas (TString name, TString upper_label)
- virtual ~QuarticCanvas ()
- void SetRunInfo (unsigned int board_id, unsigned int run_id, unsigned int spill_id, TString date)

- void SetUpperLabel (TString text)
- void FillChannel (unsigned short channel_id, double content)
- TH2D ∗ Grid ()
- void Save (TString ext="png", TString path=".")

Private Member Functions

- void Build ()
- void DrawGrid ()
- Coord GetCoordinates (unsigned short channel_id) const

Private Attributes

- TPad ∗ c1
- TPad ∗ c2
- TH2D ∗ fHist
- double fWidth
- double fHeight
- TLegend ∗ fLegend
- double fLegendX
- double fLegendY
- unsigned int fLegendNumEntries
- TPaveText ∗ fLabel1
- TPaveText ∗ fLabel2
- TPaveText ∗ fLabel3
- TPaveText ∗ fLabel4
- TString fUpperLabelText
- TPaveText ∗ fUpperLabel
- bool fLabelsDrawn
- unsigned int fBoardId
- unsigned int fRunId
- unsigned int fSpillId
- TString fRunDate

7.17.1 Detailed Description

Author

Laurent Forthomme `laurent.forthomme@cern.ch`

Date

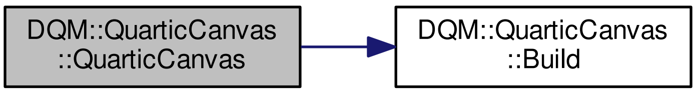
3 Aug 2015

7.17.2 Constructor & Destructor Documentation

7.17.2.1 DQM::QuarticCanvas::QuarticCanvas ( ) `[inline]`

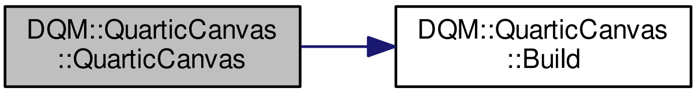
7.17.2.2 DQM::QuarticCanvas::QuarticCanvas (TString *name*, unsigned int *width* = `500`, unsigned int *height* = `500`, TString *upper_label* = `""`) `[inline]`

Here is the call graph for this function:


DQM::QuarticCanvas ::QuarticCanvas
DQM::QuarticCanvas ::Build

7.17.2.3 DQM::QuarticCanvas::QuarticCanvas (TString *name*, TString *upper_label*) `[inline]`

Here is the call graph for this function:


DQM::QuarticCanvas ::QuarticCanvas
DQM::QuarticCanvas ::Build

7.17.2.4 virtual DQM::QuarticCanvas::∼QuarticCanvas ( ) `[inline]`, `[virtual]`

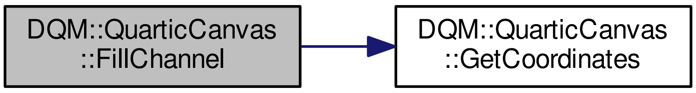
7.17.3 Member Function Documentation

7.17.3.1 void DQM::QuarticCanvas::Build ( ) `[inline]`, `[private]`

7.17.3.2 void DQM::QuarticCanvas::DrawGrid ( ) `[inline]`, `[private]`

7.17.3.3 void DQM::QuarticCanvas::FillChannel (unsigned short *channel_id*, double *content*) `[inline]`

Here is the call graph for this function:

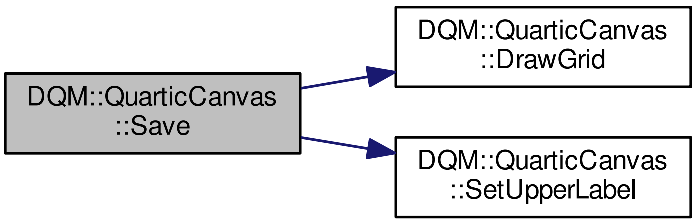

DQM::QuarticCanvas ::FillChannel
DQM::QuarticCanvas ::GetCoordinates

7.17.3.4 Coord DQM::QuarticCanvas::GetCoordinates (unsigned short *channel_id*) const [inline], [private]

7.17.3.5 TH2D∗ DQM::QuarticCanvas::Grid ( ) [inline]

7.17.3.6 void DQM::QuarticCanvas::Save (TString *ext* = "png", TString *path* = ".") [inline]

Here is the call graph for this function:


DQM::QuarticCanvas ::Save
DQM::QuarticCanvas ::DrawGrid
DQM::QuarticCanvas ::SetUpperLabel

7.17.3.7 void DQM::QuarticCanvas::SetRunInfo (unsigned int *board_id*, unsigned int *run_id*, unsigned int *spill_id*, TString *date*) [inline]

7.17.3.8 void DQM::QuarticCanvas::SetUpperLabel (TString *text*) [inline]

7.17.4 Field Documentation

7.17.4.1 TPad∗ DQM::QuarticCanvas::c1 [private]

7.17.4.2 TPad ∗ DQM::QuarticCanvas::c2 [private]

7.17.4.3 unsigned int DQM::QuarticCanvas::fBoardId [private]

7.17.4.4 double DQM::QuarticCanvas::fHeight [private]

7.17.4.5 TH2D∗ DQM::QuarticCanvas::fHist [private]

7.17.4.6 TPaveText∗ DQM::QuarticCanvas::fLabel1 [private]

7.17.4.7 TPaveText ∗ DQM::QuarticCanvas::fLabel2 [private]

7.17.4.8 TPaveText ∗ DQM::QuarticCanvas::fLabel3 [private]

7.17.4.9 TPaveText ∗ DQM::QuarticCanvas::fLabel4 [private]

7.17.4.10 bool DQM::QuarticCanvas::fLabelsDrawn [private]

7.17.4.11 TLegend∗ DQM::QuarticCanvas::fLegend [private]

7.17.4.12 unsigned int DQM::QuarticCanvas::fLegendNumEntries [private]

7.17.4.13 double DQM::QuarticCanvas::fLegendX [private]

7.17.4.14 double DQM::QuarticCanvas::fLegendY [private]

7.17.4.15 TString DQM::QuarticCanvas::fRunDate [private]

7.17.4.16 unsigned int DQM::QuarticCanvas::fRunId [private]

7.17.4.17 unsigned int DQM::QuarticCanvas::fSpillId [private]

7.17.4.18 TPaveText* DQM::QuarticCanvas::fUpperLabel [private]

7.17.4.19 TString DQM::QuarticCanvas::fUpperLabelText [private]

7.17.4.20 double DQM::QuarticCanvas::fWidth [private]

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

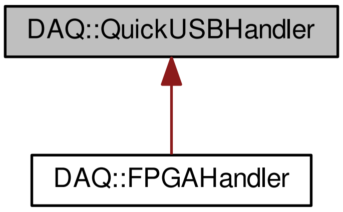
- include/QuarticCanvas.h

7.18 DAQ::QuickUSBHandler Class Reference

Generic QuickUSB communication handler.

`#include <QuickUSBHandler.h>`

Inheritance diagram for DAQ::QuickUSBHandler:



Data Structures

- struct FIFOFlags
- struct Version

Public Types

- enum HWRevision { CY7C68013AB =0x0, CY7C68013A =0x1, CY7C68013CD =0x2, CY7C68013E =0x4 }
- enum USBSpeed { USBFullSpeed =0x0, USBHighSpeed =0x1 }
- enum FPGAType { AlteraPassiveSerial =0, XilinxSlaveSerial =1 }

Public Member Functions

- QuickUSBHandler ()
- virtual ∼QuickUSBHandler ()
- void Init ()
- void Reset () const
- Version GetFWVersion () const

  Read the QuickUSB firmware revision.
- Version GetDriverVersion () const

  Read the QuickUSB driver revision.
- Version GetDLLVersion () const

  Read the QuickUSB library revision.
- void Write (uint16_t addr, uint8_t word) const

  Write a single word to the QuickUSB device.
- void Write (uint16_t addr, std::vector< uint8_t > &words, uint16_t size) const

  Write a set of words to the QuickUSB device.
- std::vector< uint8_t > Fetch (uint16_t addr, uint16_t size) const

  Receive a set of words from the QuickUSB device.
- void StartBulkTransfer (QVOIDRETURN callback(PQBULKSTREAM))
- void StopBulkTransfer ()

  Stop the data collection.

Protected Attributes

- bool fIsStopping

Private Types

- enum SettingsRegister {
  kWordWide = 0x01, kDataAddress = 0x02, kFIFOConfig = 0x03, kFPGAType = 0x04,
  kCPUConfig = 0x05, kSPIConfig = 0x06, kSlaveFIFOFlags = 0x07, kI2CTL = 0x08,
  kPortA = 0x09, kPortB = 0x0a, kPortC = 0x0b, kPortD = 0x0c,
  kPortE = 0x0d, kPortAConfig = 0x0e, kPinFlags = 0x0f, kVersionSpeed = 0x11,
  kTimeoutHigh = 0x12, kTimeoutLow = 0x13 }
- enum WordWide { k8bits =0, k16bits =1 }
- enum CPUConfig {
  kCLKOUTdisable =0x0, kCLKOUTenable =0x1, kCLKINVdisable =0x0, kCLKINVenable =0x2,
  kCLKSPD12MHz =0x0, kCLKSPD24MHz =0x4, kCLKSPD48MHz =0x8, kCLKSPDreserved =0xc,
  kUSBFullSpeedForce =0x0, kUSBFullSpeedAllow =0x8000 }
- enum SPIConfig {
  kSPIENDIANlsb =0x0, kSPIENDIANmsb =0x1, kSPICPOLnormal =0x0, kSPICPOLinverted =0x2,
  kSPICPHAsampleclock =0x0, kSPICPHAclocksample =0x4, kSPIPORTE =0x0, kSPIPORTA =0x8,
  kNCEPIN2 =0x0, kNCEPIN7 =0x10, kMISOPIN5 =0x0, kMISOPIN2 =0x20,
  kGPIFA8gpio =0x0, kGPIFA8gfiadr8 =0x8000 }
- enum I2CTL {
  kI2CBusClkSpeed100kHz =0x0, kI2CBusClkSpeed400kHZ =0x1, kHandleACK =0x0, kIgnoreACK =0x80,
  kBusError =0x600, kNoACK =0x700, kNormalCompletion =0x800, kSlaveWait =0xa00,
  kTimeout =0xb00 }
- enum LogicLevel { kLowLogic =0x0, kHighLogic =0x1 }

Private Member Functions

- void Configure () const

  Configure the board with the initial settings.
- void SetConfigRegister (SettingsRegister reg, const uint16_t &word) const

  Set a configuration register on the board.
- uint16_t GetConfigRegister (SettingsRegister reg) const

  Retrieve a single configuration register from the board.
- void SetWordWide (const WordWide &ww) const

  Set the high-speed port data width (8 or 16 bits)
- void SetDataAddress (uint16_t addr, bool increment=false, bool enable_addr_bus=false) const
- void SetFIFOConfig (uint16_t word) const
- FPGAType GetFPGAType () const

  Get the FPGA configuration scheme.
- void SetFPGAType (const FPGAType ft) const

  Set the FPGA configuration scheme.
- void SetCPUConfig (uint16_t c) const
- void SetSPIConfig (uint16_t c) const

  Configure the SPI interface.
- FIFOFlags GetSlaveFIFOFlags () const
- void SetI2CTL (uint16_t c) const
- void SetPort (const char port, const LogicLevel &lev, bool output_buf) const
- HWRevision GetHWRevision () const
- USBSpeed GetUSBSpeed () const
- uint16_t GetTimeoutHigh () const

  Return the FW timeout high (in ms)
- uint16_t GetTimeoutLow () const

  Return the FW timeout low (in ms)

Private Attributes

- std::string fDevice
- QHANDLE fHandle
- uint8_t fStreamId

Friends

- std::ostream & operator<< (std::ostream &out, const HWRevision &rev)
- std::ostream & operator<< (std::ostream &out, const USBSpeed &sp)
- std::ostream & operator<< (std::ostream &out, const FPGAType &sp)

7.18.1 Detailed Description

Generic QuickUSB communication handler.

Date

17 May 2016

Author

Laurent Forthomme `laurent.forthomme@cern.ch`

7.18.2 Member Enumeration Documentation

7.18.2.1 enum **DAQ::QuickUSBHandler::CPUConfig** `[private]`

Enumerator

kCLKOUTdisable

kCLKOUTenable

kCLKINVdisable

kCLKINVenable

kCLKSPD12MHz

kCLKSPD24MHz

kCLKSPD48MHz

kCLKSPDreserved

kUSBFullSpeedForce

kUSBFullSpeedAllow

7.18.2.2 enum **DAQ::QuickUSBHandler::FPGAType**

Enumerator

AlteraPassiveSerial

XilinxSlaveSerial

7.18.2.3 enum **DAQ::QuickUSBHandler::HWRevision**

Enumerator

CY7C68013AB

CY7C68013A

CY7C68013CD

CY7C68013E

7.18.2.4 enum **DAQ::QuickUSBHandler::I2CTL** `[private]`

Enumerator

kI2CBusClkSpeed100kHz

kI2CBusClkSpeed400kHZ

kHandleACK

kIgnoreACK

kBusError

kNoACK

kNormalCompletion

kSlaveWait

kTimeout

7.18.2.5 enum **DAQ::QuickUSBHandler::LogicLevel** `[private]`

Enumerator

kLowLogic

kHighLogic

7.18.2.6 enum **DAQ::QuickUSBHandler::SettingsRegister** `[private]`

Enumerator

kWordWide

kDataAddress

kFIFOConfig

kFPGAType

kCPUConfig

kSPIConfig

kSlaveFIFOFlags

kI2CTL

kPortA

kPortB

kPortC

kPortD

kPortE

kPortAConfig

kPinFlags

kVersionSpeed

kTimeoutHigh

kTimeoutLow

7.18.2.7 enum **DAQ::QuickUSBHandler::SPIConfig** `[private]`

Enumerator

kSPIENDIANlsb

kSPIENDIANmsb

kSPICPOLnormal

kSPICPOLinverted

kSPICPHAsampleclock

kSPICPHAclocksample

kSPIPORTE

kSPIPORTA

kNCEPIN2

kNCEPIN7

kMISOPIN5

kMISOPIN2

kGPIFA8gpio

kGPIFA8gfiadr8

7.18.2.8 enum DAQ::QuickUSBHandler::USBSpeed

Enumerator

USBFullSpeed

USBHighSpeed

7.18.2.9 enum DAQ::QuickUSBHandler::WordWide `[private]`

Enumerator

k8bits

k16bits

7.18.3 Constructor & Destructor Documentation

7.18.3.1 DAQ::QuickUSBHandler::QuickUSBHandler ( )

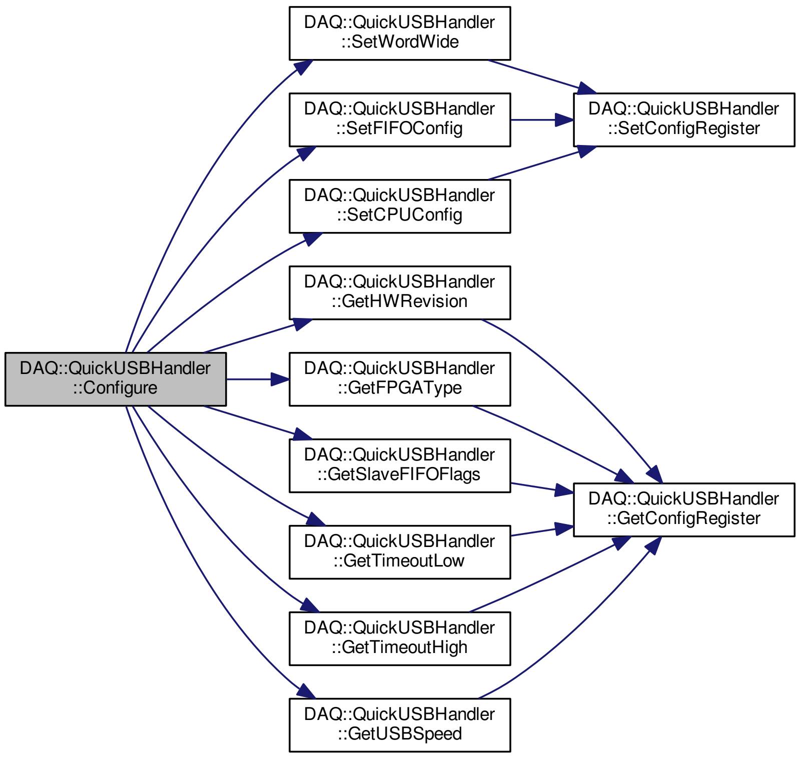
7.18.3.2 DAQ::QuickUSBHandler::∼QuickUSBHandler ( ) `[virtual]`

7.18.4 Member Function Documentation

7.18.4.1 void DAQ::QuickUSBHandler::Configure ( ) const `[private]`

Configure the board with the initial settings.

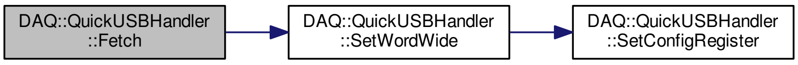
Here is the call graph for this function:


DAQ::QuickUSBHandler ::SetWordWide
DAQ::QuickUSBHandler ::SetFIFOConfig
DAQ::QuickUSBHandler ::SetConfigRegister
DAQ::QuickUSBHandler ::SetCPUConfig
DAQ::QuickUSBHandler ::GetHWRevision
DAQ::QuickUSBHandler ::Configure
DAQ::QuickUSBHandler ::GetFPGAType
DAQ::QuickUSBHandler ::GetSlaveFIFOFlags
DAQ::QuickUSBHandler ::GetConfigRegister
DAQ::QuickUSBHandler ::GetTimeoutLow
DAQ::QuickUSBHandler ::GetTimeoutHigh
DAQ::QuickUSBHandler ::GetUSBSpeed

7.18.4.2 std::vector< uint8_t > DAQ::QuickUSBHandler::Fetch (uint16_t *addr,* uint16_t *size*) const

Receive a set of words from the QuickUSB device.

Here is the call graph for this function:


DAQ::QuickUSBHandler ::Fetch
DAQ::QuickUSBHandler ::SetWordWide
DAQ::QuickUSBHandler ::SetConfigRegister

7.18.4.3 uint16_t DAQ::QuickUSBHandler::GetConfigRegister (SettingsRegister *reg*) const [inline], [private]

Retrieve a single configuration register from the board.

7.18.4.4 QuickUSBHandler::Version DAQ::QuickUSBHandler::GetDLLVersion ( ) const

Read the QuickUSB library revision.

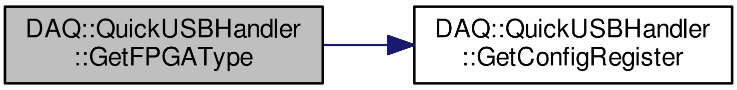
7.18.4.5 QuickUSBHandler::Version DAQ::QuickUSBHandler::GetDriverVersion ( ) const

Read the QuickUSB driver revision.

7.18.4.6 FPGAType DAQ::QuickUSBHandler::GetFPGAType ( ) const [inline], [private]

Get the FPGA configuration scheme.

Here is the call graph for this function:

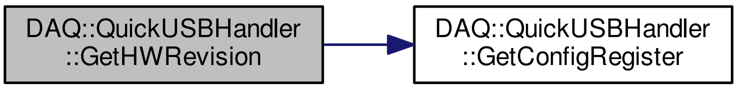

DAQ::QuickUSBHandler ::GetFPGAType
DAQ::QuickUSBHandler ::GetConfigRegister

7.18.4.7 QuickUSBHandler::Version DAQ::QuickUSBHandler::GetFWVersion ( ) const

Read the QuickUSB firmware revision.

7.18.4.8 HWRevision DAQ::QuickUSBHandler::GetHWRevision ( ) const [inline], [private]

Here is the call graph for this function:


DAQ::QuickUSBHandler ::GetHWRevision
DAQ::QuickUSBHandler ::GetConfigRegister

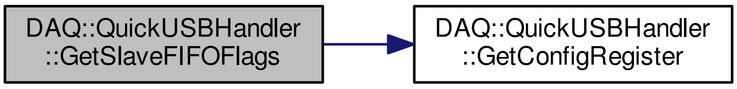
7.18.4.9 FIFOFlags DAQ::QuickUSBHandler::GetSlaveFIFOFlags ( ) const [inline], [private]

Get the slave FIFO flag status

Note

These flags are only significant when the FX2 is in slave FIFO mode

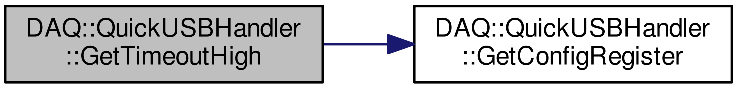
Here is the call graph for this function:


DAQ::QuickUSBHandler ::GetSlaveFIFOFlags
DAQ::QuickUSBHandler ::GetConfigRegister

7.18.4.10 uint16_t DAQ::QuickUSBHandler::GetTimeoutHigh ( ) const [inline], [private]

Return the FW timeout high (in ms)

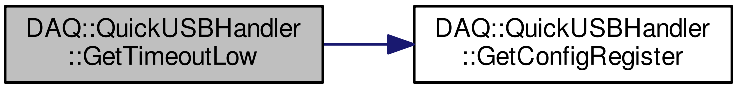
Here is the call graph for this function:


DAQ::QuickUSBHandler ::GetTimeoutHigh
DAQ::QuickUSBHandler ::GetConfigRegister

7.18.4.11 uint16_t DAQ::QuickUSBHandler::GetTimeoutLow ( ) const [inline], [private]

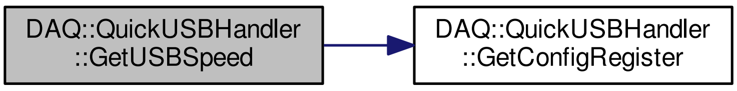
Return the FW timeout low (in ms)

Here is the call graph for this function:


DAQ::QuickUSBHandler ::GetTimeoutLow
DAQ::QuickUSBHandler ::GetConfigRegister

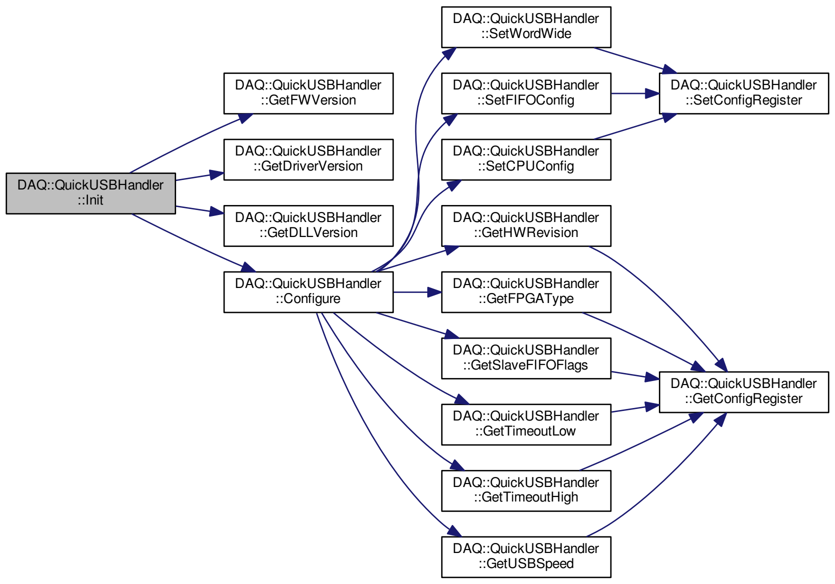
7.18.4.12 USBSpeed DAQ::QuickUSBHandler::GetUSBSpeed ( ) const [inline], [private]

Here is the call graph for this function:


DAQ::QuickUSBHandler ::GetUSBSpeed
DAQ::QuickUSBHandler ::GetConfigRegister

7.18.4.13 void DAQ::QuickUSBHandler::Init ( )

Here is the call graph for this function:



DAQ::QuickUSBHandler ::Init
DAQ::QuickUSBHandler ::GetFWVersion
DAQ::QuickUSBHandler ::GetDriverVersion
DAQ::QuickUSBHandler ::GetDLLVersion
DAQ::QuickUSBHandler ::Configure
DAQ::QuickUSBHandler ::SetWordWide
DAQ::QuickUSBHandler ::SetFIFOConfig
DAQ::QuickUSBHandler ::SetCPUConfig
DAQ::QuickUSBHandler ::GetHWRevision
DAQ::QuickUSBHandler ::GetFPGAType
DAQ::QuickUSBHandler ::GetSlaveFIFOFlags
DAQ::QuickUSBHandler ::GetTimeoutLow
DAQ::QuickUSBHandler ::GetTimeoutHigh
DAQ::QuickUSBHandler ::GetUSBSpeed
DAQ::QuickUSBHandler ::SetConfigRegister
DAQ::QuickUSBHandler ::GetConfigRegister

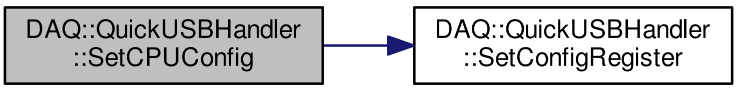
7.18.4.14 void DAQ::QuickUSBHandler::Reset ( ) const

7.18.4.15 void DAQ::QuickUSBHandler::SetConfigRegister (SettingsRegister *reg*, const uint16_t & *word*) const [inline], [private]

Set a configuration register on the board.

7.18.4.16 void DAQ::QuickUSBHandler::SetCPUConfig (uint16_t *c*) const [inline], [private]

Here is the call graph for this function:



DAQ::QuickUSBHandler ::SetCPUConfig
DAQ::QuickUSBHandler ::SetConfigRegister

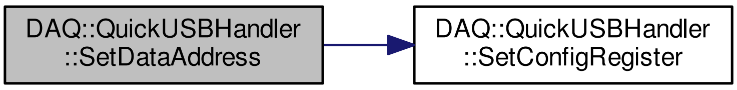
7.18.4.17 void DAQ::QuickUSBHandler::SetDataAddress (uint16_t *addr*, bool *increment* = false, bool *enable_addr_bus* = false) const [inline], [private]

Set the data bus starting address

Parameters

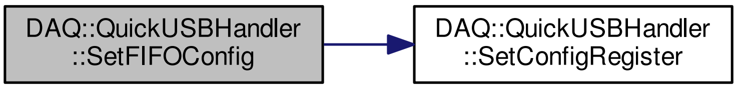
in	*increment*	Auto increment the bus address after each data transaction?
in	*enable_addr_←↩ bus*	Enable the address bus?

Here is the call graph for this function:


DAQ::QuickUSBHandler ::SetDataAddress
DAQ::QuickUSBHandler ::SetConfigRegister

7.18.4.18 void DAQ::QuickUSBHandler::SetFIFOConfig (uint16_t *word*) const [inline], [private]

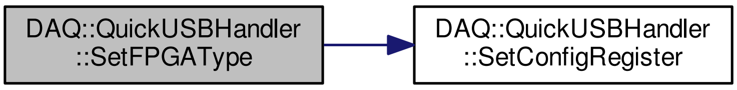
Here is the call graph for this function:


DAQ::QuickUSBHandler ::SetFIFOConfig
DAQ::QuickUSBHandler ::SetConfigRegister

7.18.4.19 void DAQ::QuickUSBHandler::SetFPGAType (const **FPGAType** *ft*) const [inline], [private]

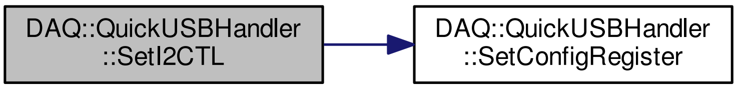
Set the FPGA configuration scheme.

Here is the call graph for this function:


DAQ::QuickUSBHandler ::SetFPGAType
DAQ::QuickUSBHandler ::SetConfigRegister

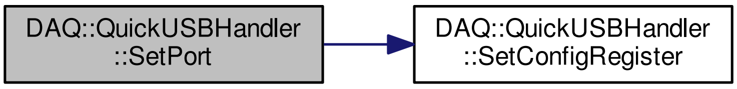
7.18.4.20 void DAQ::QuickUSBHandler::SetI2CTL (uint16_t *c*) const [inline], [private]

Here is the call graph for this function:


DAQ::QuickUSBHandler ::SetI2CTL
DAQ::QuickUSBHandler ::SetConfigRegister

7.18.4.21 void DAQ::QuickUSBHandler::SetPort (const char *port*, const LogicLevel & *lev*, bool *output_buf*) const [inline], [private]

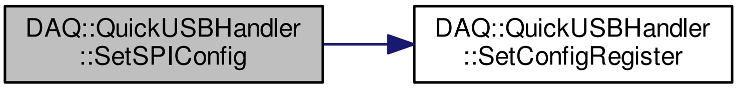
Here is the call graph for this function:


DAQ::QuickUSBHandler ::SetPort
DAQ::QuickUSBHandler ::SetConfigRegister

7.18.4.22 void DAQ::QuickUSBHandler::SetSPIConfig (uint16_t *c*) const [inline], [private]

Configure the SPI interface.

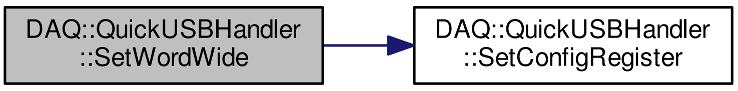
Here is the call graph for this function:


DAQ::QuickUSBHandler ::SetSPIConfig
DAQ::QuickUSBHandler ::SetConfigRegister

7.18.4.23 void DAQ::QuickUSBHandler::SetWordWide (const WordWide & *ww*) const [inline], [private]

Set the high-speed port data width (8 or 16 bits)

Here is the call graph for this function:


DAQ::QuickUSBHandler ::SetWordWide
DAQ::QuickUSBHandler ::SetConfigRegister

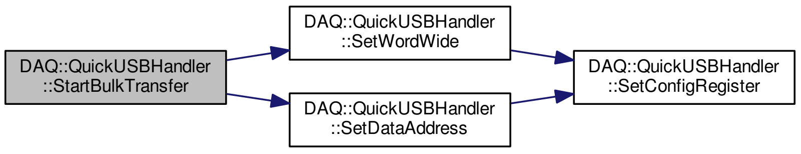
7.18.4.24 void DAQ::QuickUSBHandler::StartBulkTransfer (QVOIDRETURN *callbackPQBULKSTREAM*)

Start the data collection

Parameters

`in,out`	*callback*	Callback function called at the end of each data retrieval by a process

Here is the call graph for this function:


DAQ::QuickUSBHandler ::StartBulkTransfer
DAQ::QuickUSBHandler ::SetWordWide
DAQ::QuickUSBHandler ::SetDataAddress
DAQ::QuickUSBHandler ::SetConfigRegister

7.18.4.25 void DAQ::QuickUSBHandler::StopBulkTransfer ( )

Stop the data collection.

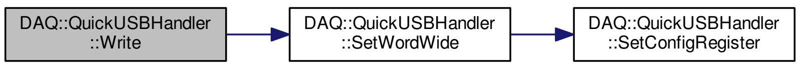
7.18.4.26 void DAQ::QuickUSBHandler::Write (uint16_t *addr,* uint8_t *word*) const `[inline]`

Write a single word to the QuickUSB device.

7.18.4.27 void DAQ::QuickUSBHandler::Write (uint16_t *addr,* std::vector< uint8_t > & *words,* uint16_t *size*) const

Write a set of words to the QuickUSB device.

Here is the call graph for this function:


DAQ::QuickUSBHandler ::Write
DAQ::QuickUSBHandler ::SetWordWide
DAQ::QuickUSBHandler ::SetConfigRegister

7.18.5 Friends And Related Function Documentation

7.18.5.1 std::ostream& operator<< (std::ostream & *out*, const **HWRevision** & *rev*) [friend]

7.18.5.2 std::ostream& operator<< (std::ostream & *out*, const **USBSpeed** & *sp*) [friend]

7.18.5.3 std::ostream& operator<< (std::ostream & *out*, const **FPGAType** & *sp*) [friend]

7.18.6 Field Documentation

7.18.6.1 std::string DAQ::QuickUSBHandler::fDevice [private]

7.18.6.2 QHANDLE DAQ::QuickUSBHandler::fHandle [private]

7.18.6.3 bool DAQ::QuickUSBHandler::fIsStopping [protected]

7.18.6.4 uint8_t DAQ::QuickUSBHandler::fStreamId [private]

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

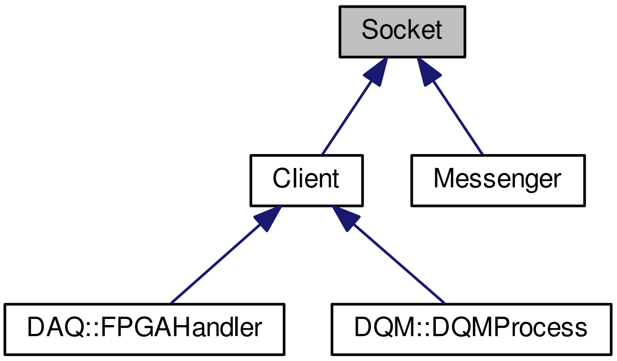
- daq/include/QuickUSBHandler.h
- daq/src/QuickUSBHandler.cpp

7.19 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, DQM, DAQ }

Type of actor playing a role on the socket.
- typedef std::set< std::pair< int, SocketType > > SocketCollection

Public Member Functions

- Socket ()
- Socket (int port)
- virtual ∼Socket ()
- void Stop ()

  Terminates the socket and all attached communications.
- void SetPort (int port)
- int GetPort () const

  Retrieve the port used for this socket.
- void AcceptConnections (Socket &socket)

  Accept connection from a client.
- void SelectConnections ()
- void SetSocketId (int sid)
- int GetSocketId () const
- SocketType GetSocketType (int sid) const
- bool IsWebSocket (int sid) const
- void DumpConnected () const

Protected Member Functions

- bool Start ()

  Start the socket.
- void Bind ()

  Bind a name to a socket.
- void PrepareConnection ()
- void Listen (int maxconn)

  Listen to incoming messages.
- void SendMessage (Message message, int id=-1) const

  Send a message on a socket.
- Message FetchMessage (int id=-1) const

  Receive a message from a socket.

Protected Attributes

- int fPort
- char fBuffer [MAX_WORD_LENGTH]
- SocketCollection fSocketsConnected
- fd_set fMaster

  Master file descriptor list.
- fd_set fReadFds

  Temp file descriptor list for select()

Private Member Functions

- void Create ()

  Create an endpoint for communication.
- void Configure ()

  Configure the socket object for communication.

Private Attributes

- int fSocketId
- struct sockaddr_in fAddress

7.19.1 Detailed Description

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

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

Author

Laurent Forthomme `laurent.forthomme@cern.ch`

Date

23 Mar 2015

7.19.2 Member Typedef Documentation

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

7.19.3 Member Enumeration Documentation

7.19.3.1 enum Socket::SocketType

Type of actor playing a role on the socket.

Enumerator

INVALID

MASTER

WEBSOCKET_CLIENT

CLIENT

DETECTOR

DQM

DAQ

7.19.4 Constructor & Destructor Documentation

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

7.19.4.2 Socket::Socket (int *port*)

7.19.4.3 Socket::∼Socket ( ) `[virtual]`

7.19.5 Member Function Documentation

7.19.5.1 void Socket::AcceptConnections (Socket & *socket*)

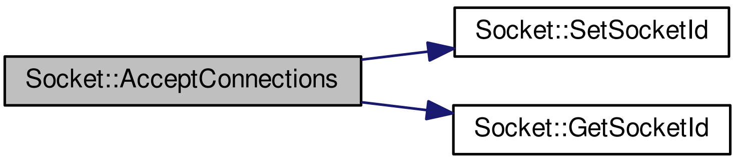
Accept connection from a client.

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

Parameters

`in,out`	*socket*	Master/client object to enable on the socket

Here is the call graph for this function:



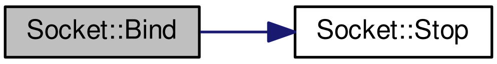
7.19.5.2 void Socket::Bind ( ) `[protected]`

Bind a name to a socket.

Returns

Success of the operation

Here is the call graph for this function:



7.19.5.3 void Socket::Configure ( ) `[private]`

Configure the socket object for communication.

7.19.5.4 void Socket::Create ( ) `[private]`

Create an endpoint for communication.

7.19.5.5 void Socket::DumpConnected ( ) const

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

Receive a message from a socket.

Returns

Received message as a std::string

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

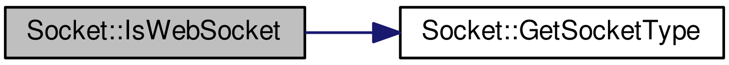
Retrieve the port used for this socket.

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

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

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

Here is the call graph for this function:

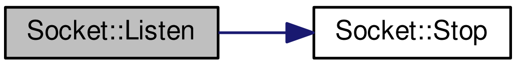


7.19.5.11 void Socket::Listen (int *maxconn*) `[protected]`

Listen to incoming messages.

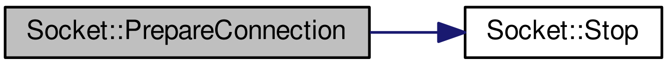
Set the socket to listen to any message coming from outside

Here is the call graph for this function:



7.19.5.12 void Socket::PrepareConnection ( ) [protected]

Here is the call graph for this function:



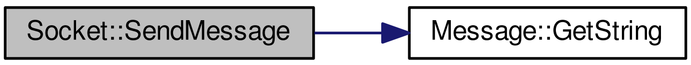
7.19.5.13 void Socket::SelectConnections ( )

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

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

Send a message on a socket.

Here is the call graph for this function:



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

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

7.19.5.17 bool Socket::Start ( ) [protected]

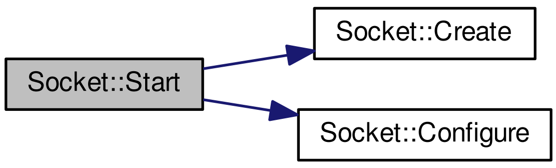
Start the socket.

Launch all mandatory operations to set the socket to be used

Returns

Success of the operation

Here is the call graph for this function:



7.19.5.18 void Socket::Stop ( )

Terminates the socket and all attached communications.

7.19.6 Field Documentation

7.19.6.1 struct sockaddr_in Socket::fAddress `[private]`

7.19.6.2 char Socket::fBuffer[MAX_WORD_LENGTH] `[protected]`

7.19.6.3 fd_set Socket::fMaster `[protected]`

Master file descriptor list.

7.19.6.4 int Socket::fPort `[protected]`

7.19.6.5 fd_set Socket::fReadFds `[protected]`

Temp file descriptor list for select()

7.19.6.6 int Socket::fSocketId `[private]`

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

7.19.6.7 SocketCollection Socket::fSocketsConnected `[protected]`

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

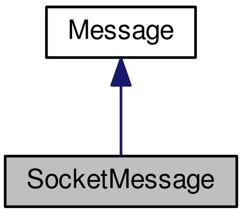
- include/Socket.h
- src/Socket.cpp

7.20 SocketMessage Class Reference

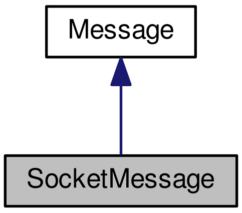
Socket-passed message type.

`#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 short value)

  Construct a socket message out of a key and a short integer-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 long value)

  Construct a socket message out of a key and a long 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, short int_value)

  Send a short integer-valued message.
- void SetKeyValue (const MessageKey &key, int int_value)

  Send an integer-valued message.
- void SetKeyValue (const MessageKey &key, long int_value)

  Send a long 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.
- std::string GetCleanedValue () const

  Extract the message's string value (without the trailing endlines)
- int GetIntValue () const

  Extract the message's integer value.
- VectorValue GetVectorValue () const

  Extract the message's vector of string value.
- void Dump (std::ostream &os=std::cout) const

Private Member Functions

- MessageMap Object () const
- std::string String () const

Private Attributes

- MessageMap fMessage

Additional Inherited Members

7.20.1 Detailed Description

Socket-passed message type.

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

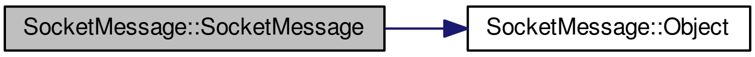
26 Mar 2015

7.20.2 Constructor & Destructor Documentation

7.20.2.1 SocketMessage::SocketMessage ( ) [inline]

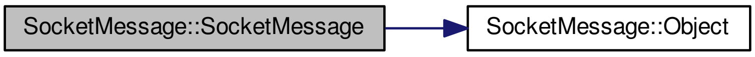
7.20.2.2 SocketMessage::SocketMessage (const Message & *msg*) [inline]

Here is the call graph for this function:



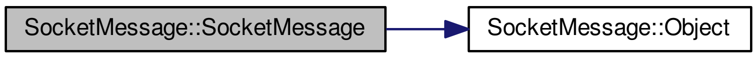
7.20.2.3 SocketMessage::SocketMessage (const char ∗ *msg_s*) [inline]

Here is the call graph for this function:



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

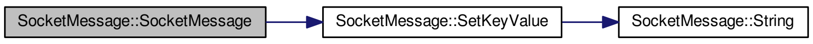
Here is the call graph for this function:



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

Construct a socket message out of a key.

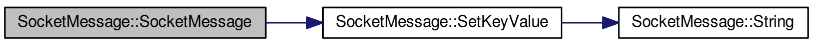
Here is the call graph for this function:



7.20.2.6 SocketMessage::SocketMessage (const MessageKey & *key,* const char ∗ *value*) [inline]

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

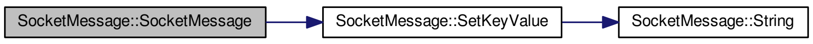
Here is the call graph for this function:



7.20.2.7 SocketMessage::SocketMessage (const MessageKey & *key,* std::string *value*) [inline]

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

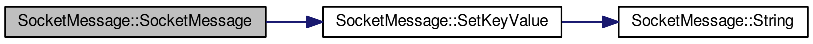
Here is the call graph for this function:



7.20.2.8 SocketMessage::SocketMessage (const MessageKey & *key,* const short *value*) [inline]

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

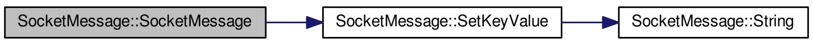
Here is the call graph for this function:



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

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

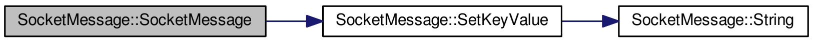
Here is the call graph for this function:



7.20.2.10 SocketMessage::SocketMessage (const MessageKey & *key,* const long *value*) [inline]

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

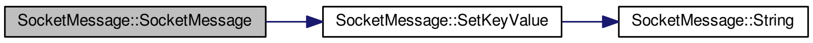
Here is the call graph for this function:



7.20.2.11 SocketMessage::SocketMessage (const MessageKey & *key,* const float *value*) [inline]

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

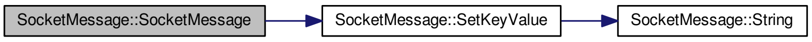
Here is the call graph for this function:



7.20.2.12 SocketMessage::SocketMessage (const MessageKey & *key,* const double *value*) [inline]

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

Here is the call graph for this function:



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

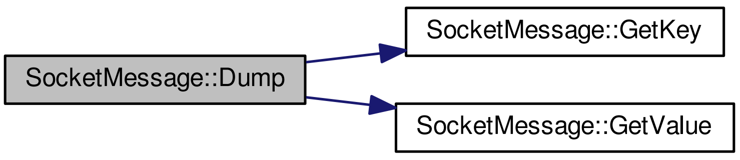
Construct a socket message out of a map of key/string-type value.

7.20.2.14 SocketMessage::∼SocketMessage ( ) [inline]

7.20.3 Member Function Documentation

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

Here is the call graph for this function:



7.20.3.2 std::string SocketMessage::GetCleanedValue ( ) const [inline]

Extract the message's string value (without the trailing endlines)

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

Extract the message's integer value.

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

Extract the message's key.

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

Extract the whole key:value message.

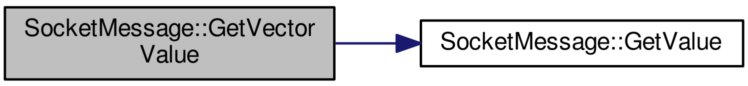
7.20.3.6 std::string SocketMessage::GetValue ( ) const [inline]

Extract the message's string value.

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

Extract the message's vector of string value.

Here is the call graph for this function:

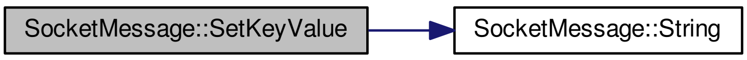


7.20.3.8 MessageMap SocketMessage::Object ( ) const [inline], [private]

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

String-valued message.

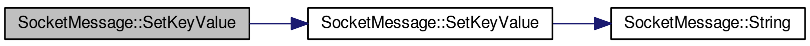
Here is the call graph for this function:



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

Send a short integer-valued message.

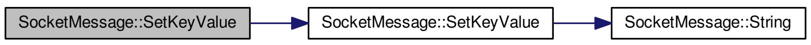
Here is the call graph for this function:



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

Send an integer-valued message.

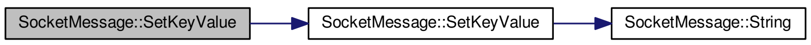
Here is the call graph for this function:



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

Send a long integer-valued message.

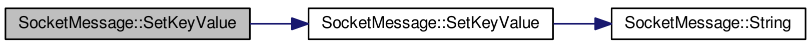
Here is the call graph for this function:



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

Float-valued message.

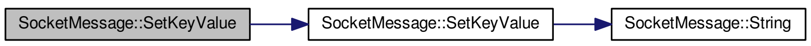
Here is the call graph for this function:



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

Double-valued message.

Here is the call graph for this function:



7.20.3.15 std::string SocketMessage::String ( ) const [inline], [private]

7.20.4 Field Documentation

7.20.4.1 MessageMap SocketMessage::fMessage [private]

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

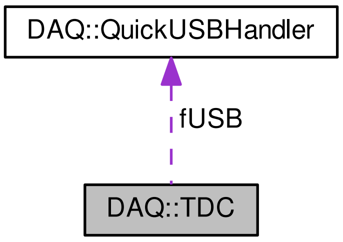
- include/SocketMessage.h

7.21 DAQ::TDC Class Reference

HPTDC object.

`#include <TDC.h>`

Collaboration diagram for DAQ::TDC:



Public Types

- enum AcquisitionMode { CONT_STORAGE, TRIG_MATCH }

    TDC acquisition mode.
- enum DetectionMode { PAIR = 0x0, OTRAILING = 0x1, OLEADING = 0x2, TRAILEAD = 0x3 }

Public Member Functions

- TDC (unsigned int id, QuickUSBHandler ∗h)
- ∼TDC ()
- void SetSetupRegister (const TDCSetup &c)

    Submit the HPTDC setup word as a TDCSetup object.
- TDCSetup GetSetupRegister ()

    Retrieve the HPTDC setup word as a TDCSetup object.
- bool CheckFirmwareVersion () const
- void SoftReset ()
- TDCEventCollection FetchEvents ()
- void ReadStatus ()

Private Member Functions

- void SendConfiguration ()

    Set the setup word to the HPTDC internal setup register.

- void ReadConfiguration ()

  Read the setup word from the HPTDC internal setup register.
- template<class T >
  void WriteRegister (unsigned int r, const T &v)

  Write one register content on the HPTDC inner memory.
- template<class T >
  T ReadRegister (unsigned int r)

  Retrieve one register content from the HPTDC inner memory.

Private Attributes

- unsigned int fId
- QuickUSBHandler ∗ fUSB
- TDCSetup fSetup
- TDCControl fControl
- TDCBoundaryScan fBS
- TDCStatus fStatus

7.21.1 Detailed Description

HPTDC object.

Author

Laurent Forthomme `laurent.forthomme@cern.ch`

Date

27 Apr 2015

7.21.2 Member Enumeration Documentation

7.21.2.1 enum DAQ::TDC::DetectionMode

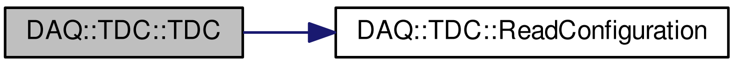
Enumerator

PAIR
OTRAILING
OLEADING
TRAILEAD

7.21.3 Constructor & Destructor Documentation

7.21.3.1 DAQ::TDC::TDC (unsigned int *id*, QuickUSBHandler ∗ *h*)

Here is the call graph for this function:



7.21.3.2 DAQ::TDC::∼TDC ( ) [inline]

7.21.4 Member Function Documentation

7.21.4.1 bool DAQ::TDC::CheckFirmwareVersion ( ) const

7.21.4.2 TDCEventCollection DAQ::TDC::FetchEvents ( )

7.21.4.3 TDCSetup DAQ::TDC::GetSetupRegister ( ) [inline]

Retrieve the HPTDC setup word as a TDCSetup object.

7.21.4.4 void DAQ::TDC::ReadConfiguration ( ) [private]

Read the setup word from the HPTDC internal setup register.

7.21.4.5 template<class T > T DAQ::TDC::ReadRegister (unsigned int *r*) [private]

Retrieve one register content from the HPTDC inner memory.

7.21.4.6 void DAQ::TDC::ReadStatus ( ) [inline]

7.21.4.7 void DAQ::TDC::SendConfiguration ( ) [private]

Set the setup word to the HPTDC internal setup register.

7.21.4.8 void DAQ::TDC::SetSetupRegister (const TDCSetup & *c*) [inline]

Submit the HPTDC setup word as a TDCSetup object.

7.21.4.9 void DAQ::TDC::SoftReset ( )

7.21.4.10 template<class T > void DAQ::TDC::WriteRegister (unsigned int *r*, const T & *v*) [private]

Write one register content on the HPTDC inner memory.

7.21.5 Field Documentation

7.21.5.1 TDCBoundaryScan DAQ::TDC::fBS [private]

7.21.5.2 TDCControl DAQ::TDC::fControl [private]

7.21.5.3 unsigned int DAQ::TDC::fId [private]

7.21.5.4 TDCSetup DAQ::TDC::fSetup [private]

7.21.5.5 TDCStatus DAQ::TDC::fStatus [private]

7.21.5.6 QuickUSBHandler∗ DAQ::TDC::fUSB [private]

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

- daq/include/TDC.h
- daq/src/TDC.cpp

7.22 OnlineDBHandler::TDCConditions Struct Reference

```
#include <OnlineDBHandler.h>
```

Public Member Functions

- bool operator== (const TDCConditions &rhs) const
- TDCConditions & operator= (const TDCConditions &rhs)

Data Fields

- unsigned int run_id
- unsigned short tdc_id
- unsigned long tdc_address
- unsigned short tdc_acq_mode
- unsigned short tdc_det_mode
- std::string detector

7.22.1 Member Function Documentation

7.22.1.1 TDCConditions& OnlineDBHandler::TDCConditions::operator= (const TDCConditions & *rhs*) `[inline]`

7.22.1.2 bool OnlineDBHandler::TDCConditions::operator== (const TDCConditions & *rhs*) const `[inline]`

7.22.2 Field Documentation

7.22.2.1 std::string OnlineDBHandler::TDCConditions::detector

7.22.2.2 unsigned int OnlineDBHandler::TDCConditions::run_id

7.22.2.3 unsigned short OnlineDBHandler::TDCConditions::tdc_acq_mode

7.22.2.4 unsigned long OnlineDBHandler::TDCConditions::tdc_address

7.22.2.5 unsigned short OnlineDBHandler::TDCConditions::tdc_det_mode

7.22.2.6 unsigned short OnlineDBHandler::TDCConditions::tdc_id

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

- include/OnlineDBHandler.h

7.23 TDCErrorFlag Class Reference

Error flags handler.

```
#include <TDCEvent.h>
```

Public Member Functions

- TDCErrorFlag (uint16_t ef)
- virtual ∼TDCErrorFlag ()
- uint16_t GetWord () const
- void Dump () const
- bool HasReadoutFIFOOverflow (unsigned int group_id) const

  Check whether hits have been lost from read-out FIFO overflow in a given group.
- bool HasL1BufferOverflow (unsigned int group_id) const

  Check whether hits have been lost from L1 buffer overflow in a given group.
- bool HasGroupError (unsigned int group_id) const

  Check whether hits have been lost due to error in a given group.
- bool HasReachedEventSizeLimit () const

  Hits rejected because of programmed event size limit.
- bool HasTriggerFIFOOverflow () const

  Event lost (trigger FIFO overflow)
- bool HasInternalChipError () const

  Internal fatal chip error has been detected.

Private Attributes

- uint16_t fWord

Friends

- std::ostream & operator<< (std::ostream &os, const TDCErrorFlag &ef)

7.23.1 Detailed Description

Error flags handler.

Author

Laurent Forthomme `laurent.forthomme@cern.ch`

Date

22 Jun 2015

7.23.2 Constructor & Destructor Documentation

7.23.2.1 TDCErrorFlag::TDCErrorFlag (uint16_t *ef*) `[inline]`

7.23.2.2 virtual TDCErrorFlag::∼TDCErrorFlag ( ) `[inline]`, `[virtual]`

7.23.3 Member Function Documentation

7.23.3.1 void TDCErrorFlag::Dump ( ) const `[inline]`

7.23.3.2 uint16_t TDCErrorFlag::GetWord ( ) const `[inline]`

7.23.3.3 bool TDCErrorFlag::HasGroupError (unsigned int *group_id*) const `[inline]`

Check whether hits have been lost due to error in a given group.

7.23.3.4 bool TDCErrorFlag::HasInternalChipError ( ) const [inline]

Internal fatal chip error has been detected.

7.23.3.5 bool TDCErrorFlag::HasL1BufferOverflow (unsigned int *group_id*) const [inline]

Check whether hits have been lost from L1 buffer overflow in a given group.

7.23.3.6 bool TDCErrorFlag::HasReachedEventSizeLimit ( ) const [inline]

Hits rejected because of programmed event size limit.

7.23.3.7 bool TDCErrorFlag::HasReadoutFIFOOverflow (unsigned int *group_id*) const [inline]

Check whether hits have been lost from read-out FIFO overflow in a given group.

7.23.3.8 bool TDCErrorFlag::HasTriggerFIFOOverflow ( ) const [inline]

Event lost (trigger FIFO overflow)

7.23.4 Friends And Related Function Documentation

7.23.4.1 std::ostream& operator<< (std::ostream & *os,* const TDCErrorFlag & *ef*) [friend]

7.23.5 Field Documentation

7.23.5.1 uint16_t TDCErrorFlag::fWord [private]

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

- include/TDCEvent.h

7.24 TDCEvent Class Reference

HPTDC event parser.

`#include <TDCEvent.h>`

Public Types

- enum EventType {
  TDCMeasurement = 0x0, TDCHeader = 0x1, TDCTrailer = 0x3, TDCError = 0x4,
  GlobalHeader = 0x8, GlobalTrailer = 0x10, ETTT = 0x11, Filler = 0x18,
  Trigger = 0x1f }

Public Member Functions

- TDCEvent ()
- TDCEvent (const TDCEvent &ev)
- TDCEvent (const uint32_t &word)
- TDCEvent (const EventType &ev)

- virtual ∼TDCEvent ()
- void Dump () const
- void SetWord (const uint32_t &word)
- uint32_t GetWord () const
- EventType GetType () const

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

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

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

    Total number of words in event (including headers and trailers)
- unsigned int GetGeo () const
- unsigned int GetChannelId () const

    Channel number for.
- uint32_t GetEventCount () const

    Total number of events.
- uint16_t GetBunchId () const

    Bunch identifier of trigger (or trigger time tag)
- bool IsTrailing () const

    Are we dealing with a trailing or a leading measurement?
- uint32_t GetETTT () const

    Extended trigger time tag.
- uint32_t GetTime (bool pair=false) const

    Edge measurement in programmed time resolution.
- unsigned int GetWidth () const

    Width of pulse in programmed time resolution.
- unsigned int GetStatus () const
- TDCErrorFlag GetErrorFlags () const

    Return error flags if an error condition has been detected.

Private Attributes

- uint32_t fWord

7.24.1 Detailed Description

HPTDC event parser.

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

Author

Laurent Forthomme `laurent.forthomme@cern.ch`

Date

4 May 2015

7.24.2 Member Enumeration Documentation

7.24.2.1 enum **TDCEvent::EventType**

Enumerator

TDCMeasurement

TDCHeader

TDCTrailer

TDCError

GlobalHeader

GlobalTrailer

ETTT

Filler

Trigger

7.24.3 Constructor & Destructor Documentation

7.24.3.1 TDCEvent::TDCEvent ( ) `[inline]`

7.24.3.2 TDCEvent::TDCEvent (const **TDCEvent** & *ev*) `[inline]`

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

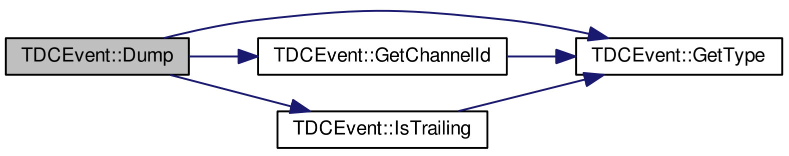
7.24.3.4 TDCEvent::TDCEvent (const **EventType** & *ev*) `[inline]`

7.24.3.5 virtual TDCEvent::∼TDCEvent ( ) `[inline]`, `[virtual]`

7.24.4 Member Function Documentation

7.24.4.1 void TDCEvent::Dump ( ) const `[inline]`

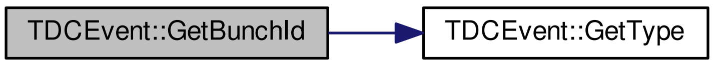
Here is the call graph for this function:



7.24.4.2 uint16_t TDCEvent::GetBunchId ( ) const `[inline]`

Bunch identifier of trigger (or trigger time tag)

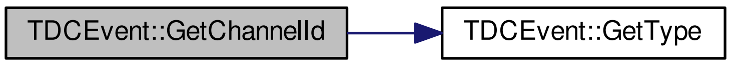
Here is the call graph for this function:



7.24.4.3 unsigned int TDCEvent::GetChannelId ( ) const [inline]

Channel number for.

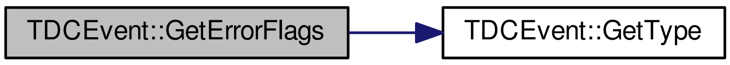
Here is the call graph for this function:



7.24.4.4 TDCErrorFlag TDCEvent::GetErrorFlags ( ) const [inline]

Return error flags if an error condition has been detected.

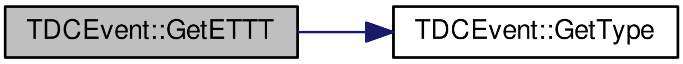
Here is the call graph for this function:



7.24.4.5 uint32_t TDCEvent::GetETTT ( ) const [inline]

Extended trigger time tag.

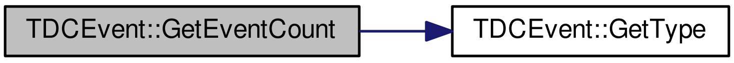
Here is the call graph for this function:



7.24.4.6 uint32_t TDCEvent::GetEventCount ( ) const [inline]

Total number of events.

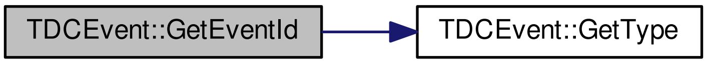
Here is the call graph for this function:



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

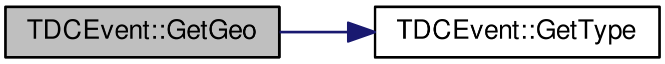
Event identifier from event counter.

Here is the call graph for this function:



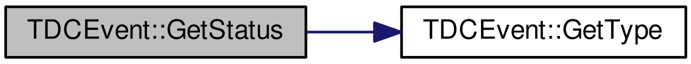
7.24.4.8 unsigned int TDCEvent::GetGeo ( ) const [inline]

Here is the call graph for this function:



7.24.4.9 unsigned int TDCEvent::GetStatus ( ) const [inline]

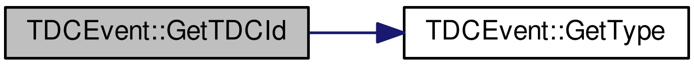
Here is the call graph for this function:



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

Programmed identifier of master TDC providing the event.

Here is the call graph for this function:



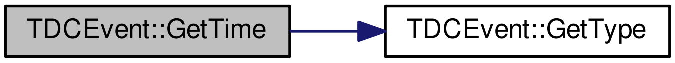
7.24.4.11 uint32_t TDCEvent::GetTime (bool *pair* = false) const [inline]

Edge measurement in programmed time resolution.

Parameters

in	*pair*	Are we dealing with a pair measurement? (only for leading time word)

Here is the call graph for this function:



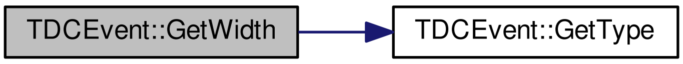
7.24.4.12 EventType TDCEvent::GetType ( ) const [inline]

Type of packet read out from the TDC.

7.24.4.13 unsigned int TDCEvent::GetWidth ( ) const [inline]

Width of pulse in programmed time resolution.

Here is the call graph for this function:

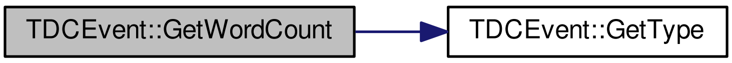


7.24.4.14 uint32_t TDCEvent::GetWord ( ) const [inline]

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

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

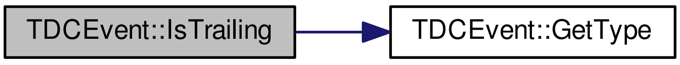
Here is the call graph for this function:



7.24.4.16 bool TDCEvent::IsTrailing ( ) const [inline]

Are we dealing with a trailing or a leading measurement?

Here is the call graph for this function:



7.24.4.17 void TDCEvent::SetWord (const uint32_t & *word*) [inline]

7.24.5 Field Documentation

7.24.5.1 uint32_t TDCEvent::fWord [private]

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

- include/TDCEvent.h

7.25 TDCMeasurement Class Reference

```
#include <TDCMeasurement.h>
```

Public Member Functions

- TDCMeasurement ()
- TDCMeasurement (const std::vector< TDCEvent > &v)
- ∼TDCMeasurement ()
- void Dump ()
- void SetEventsCollection (const std::vector< TDCEvent > &v)
- uint32_t GetLeadingTime (unsigned short event_id=0)
- uint32_t GetTrailingTime (unsigned short event_id=0)
- uint16_t GetToT (unsigned short event_id=0)
- uint16_t GetChannelId (unsigned short event_id=0)
- uint16_t GetTDCId ()
- uint16_t GetEventId ()
- uint16_t GetBunchId ()
- uint32_t GetETTT ()
- size_t NumEvents () const
- size_t NumErrors () const

Private Attributes

- std::map< TDCEvent::EventType, TDCEvent > fMap
- std::vector< std::pair< TDCEvent, TDCEvent > > fEvents

7.25.1 Detailed Description

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

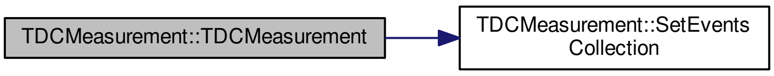
Jun 2015

7.25.2 Constructor & Destructor Documentation

7.25.2.1 TDCMeasurement::TDCMeasurement ( ) [inline]

7.25.2.2 TDCMeasurement::TDCMeasurement (const std::vector< TDCEvent > & *v*) [inline]

Here is the call graph for this function:

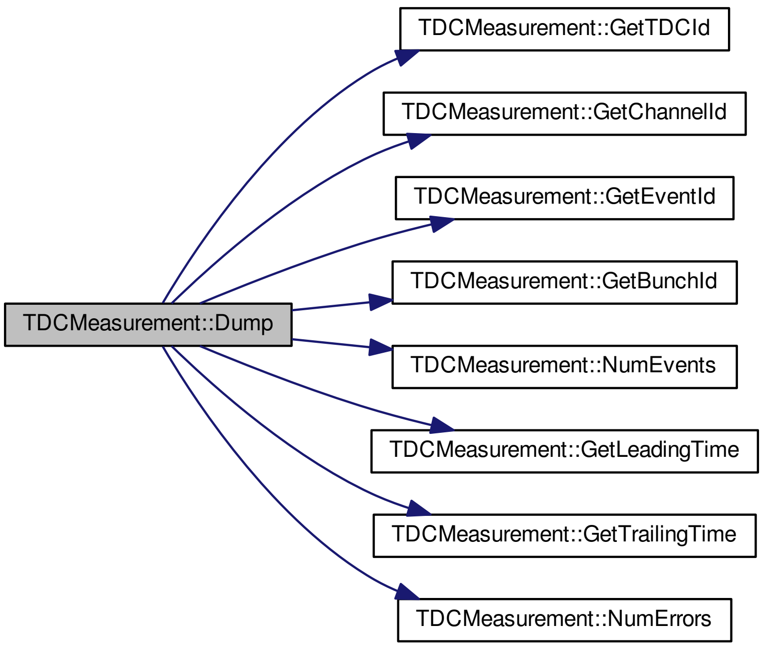

TDCMeasurement::TDCMeasurement
TDCMeasurement::SetEvents Collection

7.25.2.3 TDCMeasurement::∼TDCMeasurement ( ) [inline]

7.25.3 Member Function Documentation

7.25.3.1 void TDCMeasurement::Dump ( ) [inline]

Here is the call graph for this function:


TDCMeasurement::Dump
TDCMeasurement::GetTDCId
TDCMeasurement::GetChannelId
TDCMeasurement::GetEventId
TDCMeasurement::GetBunchId
TDCMeasurement::NumEvents
TDCMeasurement::GetLeadingTime
TDCMeasurement::GetTrailingTime
TDCMeasurement::NumErrors

7.25.3.2 uint16_t TDCMeasurement::GetBunchId ( ) [inline]

7.25.3.3 uint16_t TDCMeasurement::GetChannelId (unsigned short *event_id* = 0) [inline]

7.25.3.4 uint32_t TDCMeasurement::GetETTT ( ) [inline]

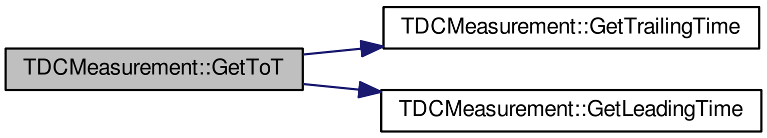
7.25.3.5 uint16_t TDCMeasurement::GetEventId ( ) [inline]

7.25.3.6 uint32_t TDCMeasurement::GetLeadingTime (unsigned short *event_id* = 0) [inline]

7.25.3.7 uint16_t TDCMeasurement::GetTDCId ( ) [inline]

7.25.3.8 uint16_t TDCMeasurement::GetToT (unsigned short *event_id* = 0) [inline]

Here is the call graph for this function:


TDCMeasurement::GetToT
TDCMeasurement::GetTrailingTime
TDCMeasurement::GetLeadingTime

7.25.3.9 uint32_t TDCMeasurement::GetTrailingTime (unsigned short *event_id* = 0) [inline]

7.25.3.10 size_t TDCMeasurement::NumErrors ( ) const [inline]

7.25.3.11 size_t TDCMeasurement::NumEvents ( ) const [inline]

7.25.3.12 void TDCMeasurement::SetEventsCollection (const std::vector< TDCEvent > & *v*) [inline]

7.25.4 Field Documentation

7.25.4.1 std::vector< std::pair<TDCEvent,TDCEvent> > TDCMeasurement::fEvents [private]

7.25.4.2 std::map<TDCEvent::EventType,TDCEvent> TDCMeasurement::fMap [private]

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

- include/TDCMeasurement.h

7.26 DAQ::QuickUSBHandler::Version Struct Reference

```
#include <QuickUSBHandler.h>
```

Data Fields

- QWORD MajorVersion
- QWORD MinorVersion
- QWORD BuildVersion

7.26.1 Field Documentation

7.26.1.1 QWORD DAQ::QuickUSBHandler::Version::BuildVersion

7.26.1.2 QWORD DAQ::QuickUSBHandler::Version::MajorVersion

7.26.1.3 QWORD DAQ::QuickUSBHandler::Version::MinorVersion

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

- daq/include/QuickUSBHandler.h

Index