

2015-2016 Test beam Run Control

Generated by Doxygen 1.8.10

Contents

1	Module Index	1
1.1	Modules	1
2	Namespace Index	3
2.1	Namespace List	3
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Data Structure Index	7
4.1	Data Structures	7
5	Module Documentation	9
5.1	Socket communication objects	9
5.1.1	Detailed Description	9
5.2	FPGA board control	10
5.2.1	Detailed Description	10
5.3	HPTDC chip control	11
5.3.1	Detailed Description	11
5.3.2	Enumeration Type Documentation	11
5.3.2.1	AcquisitionMode	11
5.3.2.2	AcquisitionMode	11
6	Namespace Documentation	13
6.1	DAQ Namespace Reference	13
6.1.1	Function Documentation	13
6.1.1.1	operator<<(std::ostream &os, const QuickUSBHandler::HWRevision &rev)	13
6.1.1.2	operator<<(std::ostream &os, const QuickUSBHandler::USBSpeed &sp)	13
6.1.1.3	operator<<(std::ostream &os, const QuickUSBHandler::FPGAType &t)	13
6.1.1.4	operator<<(std::ostream &os, const QuickUSBHandler::FIFOFlags &ff)	13
6.2	DQM Namespace Reference	13
7	Data Structure Documentation	15
7.1	OnlineDBHandler::BurstInfo Struct Reference	15

7.1.1	Field Documentation	15
7.1.1.1	burst_id	15
7.1.1.2	time_start	15
7.2	Client Class Reference	15
7.2.1	Detailed Description	17
7.2.2	Constructor & Destructor Documentation	17
7.2.2.1	Client()	17
7.2.2.2	Client(int port)	17
7.2.2.3	~Client()	18
7.2.3	Member Function Documentation	18
7.2.3.1	Announce()	18
7.2.3.2	Connect(const SocketType &type=CLIENT)	18
7.2.3.3	Disconnect()	19
7.2.3.4	GetType() const	19
7.2.3.5	ParseMessage(const SocketMessage &m)	19
7.2.3.6	Receive()	19
7.2.3.7	Receive(const MessageKey &key)	20
7.2.3.8	Send(const Message &m) const	20
7.2.3.9	Send(const Exception &e) const	21
7.2.3.10	SendAndReceive(const SocketMessage &m, const MessageKey &a) const	21
7.2.4	Field Documentation	21
7.2.4.1	fClientId	21
7.2.4.2	fIsConnected	21
7.2.4.3	fType	21
7.3	DQM::GastofCanvas::Coord Struct Reference	21
7.3.1	Field Documentation	21
7.3.1.1	x	22
7.3.1.2	y	22
7.4	DQM::QuarticCanvas::Coord Struct Reference	22
7.4.1	Field Documentation	22
7.4.1.1	x	22
7.4.1.2	y	22
7.5	DQM::DQMProcess Class Reference	22
7.5.1	Detailed Description	23
7.5.2	Member Enumeration Documentation	24
7.5.2.1	Action	24
7.5.3	Constructor & Destructor Documentation	24
7.5.3.1	DQMProcess(int port, unsigned short order=0, const char *det_type="")	24
7.5.3.2	~DQMProcess()	24
7.5.4	Member Function Documentation	24

7.5.4.1	IsInRun()	25
7.5.4.2	ParseMessage(uint32_t *board_address, std::string *filename)	25
7.5.4.3	Run(bool(*fcn)(unsigned int addr, std::string filename, std::vector< std::string > *outputs), const Action &act=NewPlot)	25
7.5.4.4	Run(bool(*fcn)(std::vector< std::string > *outputs), const Action &act=NewPlot)	26
7.5.5	Field Documentation	26
7.5.5.1	fAddressesCanProcess	26
7.5.5.2	fDetectorType	26
7.5.5.3	fOrder	26
7.5.5.4	fRunNumber	26
7.6	DAQ::QuickUSBHandler::FIFOFlags Struct Reference	26
7.6.1	Friends And Related Function Documentation	27
7.6.1.1	operator<<	27
7.6.2	Field Documentation	27
7.6.2.1	RDY0	27
7.6.2.2	RDY1	27
7.6.2.3	ReadFIFOEmpty	27
7.6.2.4	ReadFIFOFull	27
7.6.2.5	WriteFIFOEmpty	27
7.6.2.6	WriteFIFOFull	27
7.7	file_header_t Struct Reference	27
7.7.1	Detailed Description	27
7.7.2	Field Documentation	28
7.7.2.1	acq_mode	28
7.7.2.2	det_mode	28
7.7.2.3	magic	28
7.7.2.4	num_hptdc	28
7.7.2.5	run_id	28
7.7.2.6	spill_id	28
7.8	FileReader Class Reference	28
7.8.1	Detailed Description	29
7.8.2	Constructor & Destructor Documentation	29
7.8.2.1	FileReader()	29
7.8.2.2	FileReader(std::string name)	29
7.8.2.3	~FileReader()	30
7.8.3	Member Function Documentation	30
7.8.3.1	Clear()	30
7.8.3.2	Dump() const	30
7.8.3.3	GetAcquisitionMode() const	30
7.8.3.4	GetBurstId() const	30

7.8.3.5	GetDetectionMode() const	30
7.8.3.6	GetNextEvent(TDCEvent *)	30
7.8.3.7	GetNextMeasurement(unsigned int channel_id, TDCMeasurement *mc)	30
7.8.3.8	GetNumEvents() const	31
7.8.3.9	GetNumTDCs() const	31
7.8.3.10	GetRunId() const	31
7.8.3.11	IsOpen() const	31
7.8.3.12	Open(std::string name)	31
7.8.4	Field Documentation	31
7.8.4.1	fFile	31
7.8.4.2	fHeader	31
7.8.4.3	fNumEvents	31
7.8.4.4	fReadoutMode	31
7.8.4.5	fWriteTime	31
7.9	DAQ::FPGAHandler Class Reference	31
7.9.1	Detailed Description	33
7.9.2	Constructor & Destructor Documentation	33
7.9.2.1	FPGAHandler(int port, const char *dev)	33
7.9.2.2	~FPGAHandler()	34
7.9.3	Member Function Documentation	34
7.9.3.1	CloseFile()	34
7.9.3.2	ErrorState()	34
7.9.3.3	GetFilename() const	34
7.9.3.4	GetTDC(unsigned int i=0)	34
7.9.3.5	GetTDCControl() const	34
7.9.3.6	GetTDCSetup() const	34
7.9.3.7	GetTDCStatus() const	34
7.9.3.8	GetType() const	35
7.9.3.9	OpenFile()	35
7.9.3.10	RegisterTest() const	35
7.9.3.11	RetrieveSetupWord()	35
7.9.3.12	SendSetupWord() const	35
7.9.3.13	SetTDCSetup(const TDCSetup &s)	36
7.9.3.14	StartAcquisition()	36
7.9.3.15	Stop()	36
7.9.3.16	StopAcquisition()	36
7.9.4	Field Documentation	36
7.9.4.1	fFilename	36
7.9.4.2	fIsFileOpen	36
7.9.4.3	fIsTDCInReadout	36

7.9.4.4	fOutput	36
7.9.4.5	fSetupReg	36
7.9.4.6	fTDC	37
7.10	DQM::GastofCanvas Class Reference	37
7.10.1	Detailed Description	38
7.10.2	Constructor & Destructor Documentation	38
7.10.2.1	GastofCanvas()	38
7.10.2.2	GastofCanvas(TString name, unsigned int width=500, unsigned int height=500, TString upper_label="")	39
7.10.2.3	GastofCanvas(TString name, TString upper_label)	39
7.10.2.4	~GastofCanvas()	39
7.10.3	Member Function Documentation	39
7.10.3.1	Build()	39
7.10.3.2	DrawGrid()	39
7.10.3.3	FillChannel(unsigned short nino_id, unsigned short channel_id, double content)	39
7.10.3.4	GetCoordinates(unsigned short nino_id, unsigned short channel_id) const	40
7.10.3.5	Grid()	40
7.10.3.6	Save(TString ext="png", TString path="")	40
7.10.3.7	SetRunInfo(unsigned int board_id, unsigned int run_id, unsigned int spill_id, TString date)	40
7.10.3.8	SetUpperLabel(TString text)	40
7.10.4	Field Documentation	40
7.10.4.1	c1	40
7.10.4.2	c2	40
7.10.4.3	fBoardId	40
7.10.4.4	fHeight	40
7.10.4.5	fHist	40
7.10.4.6	fLabel1	40
7.10.4.7	fLabel2	40
7.10.4.8	fLabel3	40
7.10.4.9	fLabel4	40
7.10.4.10	fLabelsDrawn	40
7.10.4.11	fLegend	40
7.10.4.12	fLegendNumEntries	40
7.10.4.13	fLegendX	41
7.10.4.14	fLegendY	41
7.10.4.15	fRunDate	41
7.10.4.16	fRunId	41
7.10.4.17	fSpillId	41
7.10.4.18	fUpperLabel	41

7.10.4.19 fUpperLabelText	41
7.10.4.20 fWidth	41
7.11 Logger Class Reference	41
7.11.1 Detailed Description	41
7.11.2 Constructor & Destructor Documentation	41
7.11.2.1 Logger(std::ostream &lhs, std::ostream &rhs=std::cout)	41
7.11.2.2 ~Logger()	41
7.11.3 Field Documentation	41
7.11.3.1 fBuffer	41
7.11.3.2 fStream	41
7.12 LogRedirector Class Reference	42
7.12.1 Detailed Description	42
7.12.2 Constructor & Destructor Documentation	42
7.12.2.1 LogRedirector(std::ostream &stm=std::cout)	42
7.12.3 Member Function Documentation	42
7.12.3.1 contents() const	43
7.12.4 Field Documentation	43
7.12.4.1 fRedirect	43
7.12.4.2 fSS	43
7.13 Message Class Reference	43
7.13.1 Detailed Description	44
7.13.2 Constructor & Destructor Documentation	44
7.13.2.1 Message()	44
7.13.2.2 Message(const char *msg)	44
7.13.2.3 Message(std::string msg)	44
7.13.2.4 ~Message()	44
7.13.3 Member Function Documentation	44
7.13.3.1 Dump(std::ostream &os=std::cout) const	44
7.13.3.2 GetKey() const	44
7.13.3.3 GetString() const	44
7.13.3.4 IsFromWeb() const	44
7.13.4 Field Documentation	44
7.13.4.1 fString	44
7.14 Messenger Class Reference	45
7.14.1 Detailed Description	46
7.14.2 Constructor & Destructor Documentation	46
7.14.2.1 Messenger()	46
7.14.2.2 Messenger(int port)	47
7.14.2.3 ~Messenger()	47
7.14.3 Member Function Documentation	47

7.14.3.1	AddClient()	47
7.14.3.2	Broadcast(const Message &m) const	47
7.14.3.3	Connect()	48
7.14.3.4	Disconnect()	48
7.14.3.5	DisconnectClient(int sid, MessageKey key, bool force=false)	48
7.14.3.6	GetType() const	50
7.14.3.7	ProcessMessage(SocketMessage m, int sid)	50
7.14.3.8	Receive()	50
7.14.3.9	Send(const Message &m, int sid) const	51
7.14.3.10	SendAll(const Socket::SocketType &type, const Message &m) const	51
7.14.3.11	SendAll(const Socket::SocketType &type, const Exception &e) const	52
7.14.3.12	StartAcquisition()	52
7.14.3.13	StopAcquisition()	52
7.14.3.14	SwitchClientType(int sid, Socket::SocketType type)	53
7.14.4	Field Documentation	53
7.14.4.1	fNumAttempts	53
7.14.4.2	fPID	53
7.14.4.3	fStderrPipe	53
7.14.4.4	fStdoutPipe	53
7.15	OnlineDBHandler Class Reference	53
7.15.1	Detailed Description	54
7.15.2	Member Typedef Documentation	54
7.15.2.1	BurstInfos	54
7.15.2.2	RunCollection	54
7.15.2.3	TDCCConditionsCollection	54
7.15.3	Constructor & Destructor Documentation	54
7.15.3.1	OnlineDBHandler(std::string path=std::string(std::getenv("PPS_PATH"))+"/run← _infos.db")	54
7.15.3.2	~OnlineDBHandler()	55
7.15.4	Member Function Documentation	55
7.15.4.1	BuildTables()	55
7.15.4.2	GetLastBurst(unsigned int run) const	55
7.15.4.3	GetLastRun() const	55
7.15.4.4	GetRunInfo(unsigned int run) const	55
7.15.4.5	GetRuns() const	55
7.15.4.6	GetTDCCConditions(unsigned int run_id) const	55
7.15.4.7	NewBurst()	55
7.15.4.8	NewRun()	55
7.15.4.9	Select(std::string req, int num_fields=-1) const	55
7.15.4.10	SetHVConditions(unsigned short channel_id, unsigned int vmax, unsigned imax)	55

7.15.4.11 SetTDCConditions(unsigned short tdc_id, unsigned long tdc_address, unsigned short tdc_acq_mode, unsigned short tdc_det_mode, std::string detector)	56
7.15.5 Field Documentation	56
7.15.5.1 fDB	56
7.16 DQM::PPSCanvas Class Reference	56
7.16.1 Detailed Description	58
7.16.2 Constructor & Destructor Documentation	58
7.16.2.1 PPSCanvas()	58
7.16.2.2 PPSCanvas(TString name, unsigned int width=500, unsigned int height=500, TString upper_label="")	58
7.16.2.3 PPSCanvas(TString name, TString upper_label)	58
7.16.2.4 ~PPSCanvas()	58
7.16.3 Member Function Documentation	58
7.16.3.1 Build()	58
7.16.3.2 DrawGrid()	59
7.16.3.3 Grid()	59
7.16.3.4 Save(TString ext="png", TString path="")	59
7.16.3.5 SetRunInfo(unsigned int run_id, TString date)	59
7.16.3.6 SetUpperLabel(TString text)	59
7.16.4 Field Documentation	59
7.16.4.1 c1	59
7.16.4.2 c2	59
7.16.4.3 fHeight	59
7.16.4.4 fLabel1	59
7.16.4.5 fLabel2	59
7.16.4.6 fLabel3	59
7.16.4.7 fLabelsDrawn	59
7.16.4.8 fLegend	59
7.16.4.9 fLegendNumEntries	59
7.16.4.10 fLegendX	59
7.16.4.11 fLegendY	59
7.16.4.12 fRunDate	59
7.16.4.13 fRunId	59
7.16.4.14 fUpperLabel	59
7.16.4.15 fUpperLabelText	59
7.16.4.16 fWidth	60
7.17 DQM::QuarticCanvas Class Reference	60
7.17.1 Detailed Description	61
7.17.2 Constructor & Destructor Documentation	61
7.17.2.1 QuarticCanvas()	61

7.17.2.2	QuarticCanvas(TString name, unsigned int width=500, unsigned int height=500, TString upper_label="")	62
7.17.2.3	QuarticCanvas(TString name, TString upper_label)	62
7.17.2.4	~QuarticCanvas()	62
7.17.3	Member Function Documentation	62
7.17.3.1	Build()	62
7.17.3.2	DrawGrid()	62
7.17.3.3	FillChannel(unsigned short channel_id, double content)	62
7.17.3.4	GetCoordinates(unsigned short channel_id) const	63
7.17.3.5	Grid()	63
7.17.3.6	Save(TString ext="png", TString path="")	63
7.17.3.7	SetRunInfo(unsigned int board_id, unsigned int run_id, unsigned int spill_id, TString date)	63
7.17.3.8	SetUpperLabel(TString text)	63
7.17.4	Field Documentation	63
7.17.4.1	c1	63
7.17.4.2	c2	63
7.17.4.3	fBoardId	63
7.17.4.4	fHeight	63
7.17.4.5	fHist	63
7.17.4.6	fLabel1	63
7.17.4.7	fLabel2	63
7.17.4.8	fLabel3	63
7.17.4.9	fLabel4	63
7.17.4.10	fLabelsDrawn	63
7.17.4.11	fLegend	63
7.17.4.12	fLegendNumEntries	63
7.17.4.13	fLegendX	64
7.17.4.14	fLegendY	64
7.17.4.15	fRunDate	64
7.17.4.16	fRunId	64
7.17.4.17	fSpillId	64
7.17.4.18	fUpperLabel	64
7.17.4.19	fUpperLabelText	64
7.17.4.20	fWidth	64
7.18	DAQ::QuickUSBHandler Class Reference	64
7.18.1	Detailed Description	66
7.18.2	Member Enumeration Documentation	67
7.18.2.1	CPUConfig	67
7.18.2.2	FPGAType	67

7.18.2.3	HWRevision	67
7.18.2.4	I2CTL	67
7.18.2.5	LogicLevel	68
7.18.2.6	SettingsRegister	68
7.18.2.7	SPIConfig	68
7.18.2.8	USBSpeed	69
7.18.2.9	WordWide	69
7.18.3	Constructor & Destructor Documentation	69
7.18.3.1	QuickUSBHandler()	69
7.18.3.2	~QuickUSBHandler()	69
7.18.4	Member Function Documentation	69
7.18.4.1	Configure() const	69
7.18.4.2	Fetch(uint16_t addr, uint16_t size) const	70
7.18.4.3	GetConfigRegister(SettingsRegister reg) const	70
7.18.4.4	GetDLLVersion() const	70
7.18.4.5	GetDriverVersion() const	70
7.18.4.6	GetFPGAType() const	70
7.18.4.7	GetFWVersion() const	70
7.18.4.8	GetHWRevision() const	71
7.18.4.9	GetSlaveFIFOFlags() const	71
7.18.4.10	GetTimeoutHigh() const	71
7.18.4.11	GetTimeoutLow() const	72
7.18.4.12	GetUSBSpeed() const	72
7.18.4.13	Init()	73
7.18.4.14	Reset() const	73
7.18.4.15	SetConfigRegister(SettingsRegister reg, const uint16_t &word) const	73
7.18.4.16	SetCPUConfig(uint16_t c) const	73
7.18.4.17	SetDataAddress(uint16_t addr, bool increment=false, bool enable_addr_↔ bus=false) const	73
7.18.4.18	SetFIFOConfig(uint16_t word) const	74
7.18.4.19	SetFPGAType(const FPGAType ft) const	74
7.18.4.20	SetI2CTL(uint16_t c) const	75
7.18.4.21	SetPort(const char port, const LogicLevel &lev, bool output_buf) const	75
7.18.4.22	SetSPIConfig(uint16_t c) const	75
7.18.4.23	SetWordWide(const WordWide &ww) const	75
7.18.4.24	StartBulkTransfer(QVOIDRETURN callback(PQBULKSTREAM))	76
7.18.4.25	StopBulkTransfer()	76
7.18.4.26	Write(uint16_t addr, uint8_t word) const	76
7.18.4.27	Write(uint16_t addr, std::vector< uint8_t > &words, uint16_t size) const	76
7.18.5	Friends And Related Function Documentation	77

7.18.5.1	operator<<	77
7.18.5.2	operator<<	77
7.18.5.3	operator<<	77
7.18.6	Field Documentation	77
7.18.6.1	fDevice	77
7.18.6.2	fHandle	77
7.18.6.3	flsStopping	77
7.18.6.4	fStreamId	77
7.19	Socket Class Reference	77
7.19.1	Detailed Description	79
7.19.2	Member Typedef Documentation	79
7.19.2.1	SocketCollection	79
7.19.3	Member Enumeration Documentation	79
7.19.3.1	SocketType	79
7.19.4	Constructor & Destructor Documentation	79
7.19.4.1	Socket()	79
7.19.4.2	Socket(int port)	79
7.19.4.3	~Socket()	79
7.19.5	Member Function Documentation	79
7.19.5.1	AcceptConnections(Socket &socket)	79
7.19.5.2	Bind()	80
7.19.5.3	Configure()	80
7.19.5.4	Create()	80
7.19.5.5	DumpConnected() const	80
7.19.5.6	FetchMessage(int id=-1) const	80
7.19.5.7	GetPort() const	81
7.19.5.8	GetSocketId() const	81
7.19.5.9	GetSocketType(int sid) const	81
7.19.5.10	IsWebSocket(int sid) const	81
7.19.5.11	Listen(int maxconn)	81
7.19.5.12	PrepareConnection()	82
7.19.5.13	SelectConnections()	82
7.19.5.14	SendMessage(Message message, int id=-1) const	82
7.19.5.15	SetPort(int port)	82
7.19.5.16	SetSocketId(int sid)	82
7.19.5.17	Start()	82
7.19.5.18	Stop()	83
7.19.6	Field Documentation	83
7.19.6.1	fAddress	83
7.19.6.2	fBuffer	83

7.19.6.3	fMaster	83
7.19.6.4	fPort	83
7.19.6.5	fReadFds	83
7.19.6.6	fSocketId	83
7.19.6.7	fSocketsConnected	83
7.20	SocketMessage Class Reference	84
7.20.1	Detailed Description	86
7.20.2	Constructor & Destructor Documentation	86
7.20.2.1	SocketMessage()	86
7.20.2.2	SocketMessage(const Message &msg)	86
7.20.2.3	SocketMessage(const char *msg_s)	86
7.20.2.4	SocketMessage(std::string msg_s)	87
7.20.2.5	SocketMessage(const MessageKey &key)	87
7.20.2.6	SocketMessage(const MessageKey &key, const char *value)	87
7.20.2.7	SocketMessage(const MessageKey &key, std::string value)	87
7.20.2.8	SocketMessage(const MessageKey &key, const short value)	88
7.20.2.9	SocketMessage(const MessageKey &key, const int value)	88
7.20.2.10	SocketMessage(const MessageKey &key, const long value)	88
7.20.2.11	SocketMessage(const MessageKey &key, const float value)	88
7.20.2.12	SocketMessage(const MessageKey &key, const double value)	89
7.20.2.13	SocketMessage(MessageMap msg_m)	89
7.20.2.14	~SocketMessage()	89
7.20.3	Member Function Documentation	89
7.20.3.1	Dump(std::ostream &os=std::cout) const	89
7.20.3.2	GetCleanedValue() const	89
7.20.3.3	GetIntValue() const	89
7.20.3.4	GetKey() const	89
7.20.3.5	GetString() const	89
7.20.3.6	GetValue() const	90
7.20.3.7	GetVectorValue() const	90
7.20.3.8	Object() const	90
7.20.3.9	SetKeyValue(const MessageKey &key, const char *value)	90
7.20.3.10	SetKeyValue(const MessageKey &key, short int_value)	90
7.20.3.11	SetKeyValue(const MessageKey &key, int int_value)	90
7.20.3.12	SetKeyValue(const MessageKey &key, long int_value)	91
7.20.3.13	SetKeyValue(const MessageKey &key, float float_value)	91
7.20.3.14	SetKeyValue(const MessageKey &key, double double_value)	91
7.20.3.15	String() const	91
7.20.4	Field Documentation	91
7.20.4.1	fMessage	92

7.21	DAQ::TDC Class Reference	92
7.21.1	Detailed Description	93
7.21.2	Member Enumeration Documentation	93
7.21.2.1	DetectionMode	93
7.21.3	Constructor & Destructor Documentation	93
7.21.3.1	TDC(unsigned int id, QuickUSBHandler *h)	93
7.21.3.2	~TDC()	94
7.21.4	Member Function Documentation	94
7.21.4.1	CheckFirmwareVersion() const	94
7.21.4.2	FetchEvents()	94
7.21.4.3	GetSetupRegister()	94
7.21.4.4	ReadConfiguration()	94
7.21.4.5	ReadRegister(unsigned int r)	94
7.21.4.6	ReadStatus()	94
7.21.4.7	SendConfiguration()	94
7.21.4.8	SetSetupRegister(const TDCSetup &c)	94
7.21.4.9	SoftReset()	94
7.21.4.10	WriteRegister(unsigned int r, const T &v)	94
7.21.5	Field Documentation	94
7.21.5.1	fBS	94
7.21.5.2	fControl	94
7.21.5.3	fId	94
7.21.5.4	fSetup	94
7.21.5.5	fStatus	94
7.21.5.6	fUSB	94
7.22	OnlineDBHandler::TDCCConditions Struct Reference	95
7.22.1	Member Function Documentation	95
7.22.1.1	operator=(const TDCCConditions &rhs)	95
7.22.1.2	operator==(const TDCCConditions &rhs) const	95
7.22.2	Field Documentation	95
7.22.2.1	detector	95
7.22.2.2	run_id	95
7.22.2.3	tdc_acq_mode	95
7.22.2.4	tdc_address	95
7.22.2.5	tdc_det_mode	95
7.22.2.6	tdc_id	95
7.23	TDCErrorFlag Class Reference	95
7.23.1	Detailed Description	96
7.23.2	Constructor & Destructor Documentation	96
7.23.2.1	TDCErrorFlag(uint16_t ef)	96

7.23.2.2	~TDCErrorFlag()	96
7.23.3	Member Function Documentation	96
7.23.3.1	Dump() const	96
7.23.3.2	GetWord() const	96
7.23.3.3	HasGroupError(unsigned int group_id) const	96
7.23.3.4	HasInternalChipError() const	97
7.23.3.5	HasL1BufferOverflow(unsigned int group_id) const	97
7.23.3.6	HasReachedEventSizeLimit() const	97
7.23.3.7	HasReadoutFIFOOverflow(unsigned int group_id) const	97
7.23.3.8	HasTriggerFIFOOverflow() const	97
7.23.4	Friends And Related Function Documentation	97
7.23.4.1	operator<<	97
7.23.5	Field Documentation	97
7.23.5.1	fWord	97
7.24	TDCEvent Class Reference	97
7.24.1	Detailed Description	98
7.24.2	Member Enumeration Documentation	99
7.24.2.1	EventType	99
7.24.3	Constructor & Destructor Documentation	99
7.24.3.1	TDCEvent()	99
7.24.3.2	TDCEvent(const TDCEvent &ev)	99
7.24.3.3	TDCEvent(const uint32_t &word)	99
7.24.3.4	TDCEvent(const EventType &ev)	99
7.24.3.5	~TDCEvent()	99
7.24.4	Member Function Documentation	99
7.24.4.1	Dump() const	99
7.24.4.2	GetBunchId() const	99
7.24.4.3	GetChannelId() const	100
7.24.4.4	GetErrorFlags() const	100
7.24.4.5	GetETTT() const	100
7.24.4.6	GetEventCount() const	101
7.24.4.7	GetEventId() const	101
7.24.4.8	GetGeo() const	102
7.24.4.9	GetStatus() const	102
7.24.4.10	GetTDCId() const	102
7.24.4.11	GetTime(bool pair=false) const	102
7.24.4.12	GetType() const	103
7.24.4.13	GetWidth() const	103
7.24.4.14	GetWord() const	103
7.24.4.15	GetWordCount() const	103

7.24.4.16 IsTrailing() const	104
7.24.4.17 SetWord(const uint32_t &word)	104
7.24.5 Field Documentation	104
7.24.5.1 fWord	104
7.25 TDCMeasurement Class Reference	104
7.25.1 Detailed Description	105
7.25.2 Constructor & Destructor Documentation	105
7.25.2.1 TDCMeasurement()	105
7.25.2.2 TDCMeasurement(const std::vector< TDCEvent > &v)	105
7.25.2.3 ~TDCMeasurement()	105
7.25.3 Member Function Documentation	105
7.25.3.1 Dump()	106
7.25.3.2 GetBunchId()	106
7.25.3.3 GetChannelId(unsigned short event_id=0)	106
7.25.3.4 GetETTT()	106
7.25.3.5 GetEventId()	106
7.25.3.6 GetLeadingTime(unsigned short event_id=0)	106
7.25.3.7 GetTDCId()	106
7.25.3.8 GetToT(unsigned short event_id=0)	107
7.25.3.9 GetTrailingTime(unsigned short event_id=0)	107
7.25.3.10 NumErrors() const	107
7.25.3.11 NumEvents() const	107
7.25.3.12 SetEventsCollection(const std::vector< TDCEvent > &v)	107
7.25.4 Field Documentation	107
7.25.4.1 fEvents	107
7.25.4.2 fMap	107
7.26 DAQ::QuickUSBHandler::Version Struct Reference	107
7.26.1 Field Documentation	107
7.26.1.1 BuildVersion	107
7.26.1.2 MajorVersion	107
7.26.1.3 MinorVersion	107

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Socket communication objects	9
FPGA board control	10
HPTDC chip control	11

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

DAQ	13
DQM	13

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

OnlineDBHandler::BurstInfo	15
DQM::GastofCanvas::Coord	21
DQM::QuarticCanvas::Coord	22
DAQ::QuickUSBHandler::FIFOFlags	26
file_header_t	27
FileReader	28
Logger	41
LogRedirector	42
Message	43
SocketMessage	84
OnlineDBHandler	53
DAQ::QuickUSBHandler	64
DAQ::FPGAHandler	31
Socket	77
Client	15
DAQ::FPGAHandler	31
DQM::DQMProcess	22
Messenger	45
TCanvas	
DQM::GastofCanvas	37
DQM::PPSCanvas	56
DQM::QuarticCanvas	60
DAQ::TDC	92
OnlineDBHandler::TDCConditions	95
TDCErrorFlag	95
TDCEvent	97
TDCMeasurement	104
DAQ::QuickUSBHandler::Version	107

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

OnlineDBHandler::BurstInfo	15
Client	
Base client object for the socket	15
DQM::GastofCanvas::Coord	21
DQM::QuarticCanvas::Coord	22
DQM::DQMProcess	
Handler for a common DQM process to run on the socket	22
DAQ::QuickUSBHandler::FIFOFlags	26
file_header_t	
Header to the output files	27
FileReader	
Handler for a TDC output file readout	28
DAQ::FPGAHandler	
Driver for timing detectors' FPGA readout	31
DQM::GastofCanvas	37
Logger	
Redirect outputs to another output stream	41
LogRedirector	
Redirect output stream to a string	42
Message	
Base socket message type	43
Messenger	
Base master object for the socket	45
OnlineDBHandler	
Handler for the run information online database	53
DQM::PPSCanvas	56
DQM::QuarticCanvas	60
DAQ::QuickUSBHandler	
Generic QuickUSB communication handler	64
Socket	
Base socket object from which clients/master from a socket inherit	77
SocketMessage	
Socket-passed message type	84
DAQ::TDC	
HPTDC object	92
OnlineDBHandler::TDCConditions	95
TDCErrorFlag	
Error flags handler	95

TDCEvent	
HPTDC event parser	97
TDCMeasurement	104
DAQ::QuickUSBHandler::Version	107

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:

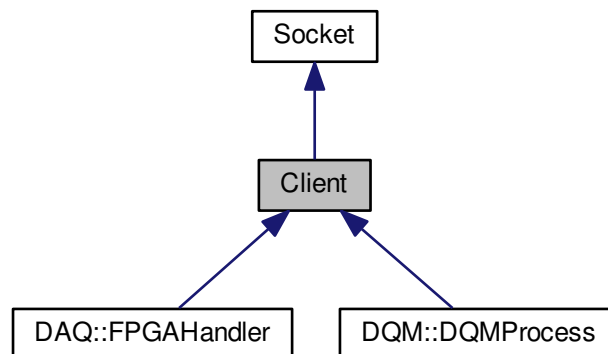
- include/OnlineDBHandler.h

7.2 Client Class Reference

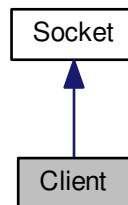
Base client object for the socket.

```
#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]

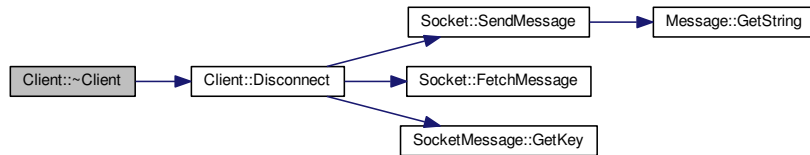
General void client constructor.

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:

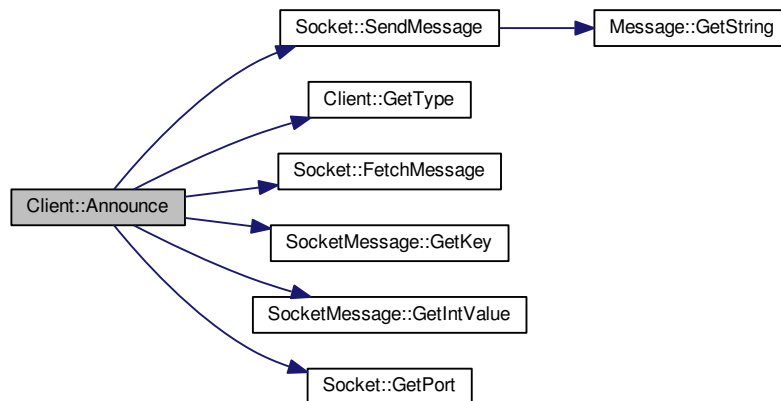


7.2.3 Member Function Documentation

7.2.3.1 void Client::Announce () [private]

Announce our entry on the socket to its master.

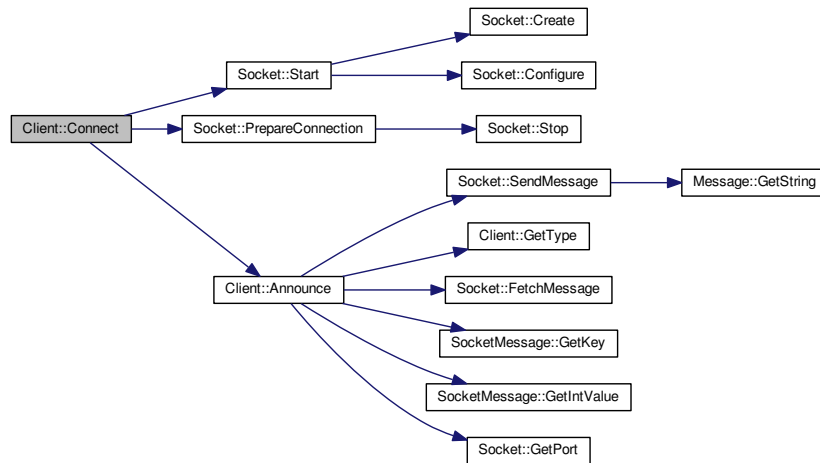
Here is the call graph for this function:



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

Bind this client to the socket.

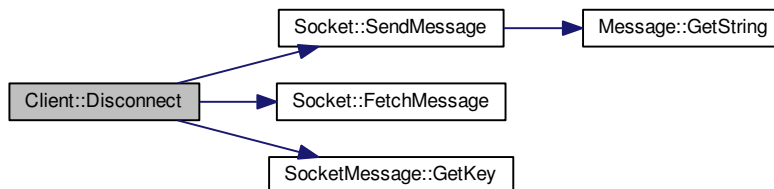
Here is the call graph for this function:



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](#).

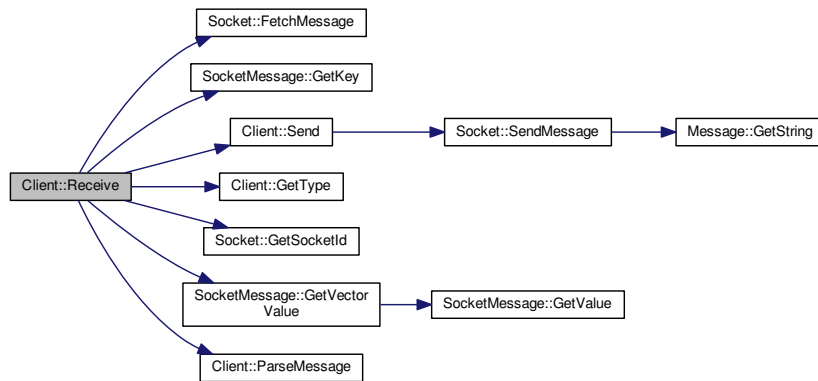
7.2.3.5 virtual void Client::ParseMessage (const SocketMessage & m) [inline],[virtual]

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

7.2.3.6 void Client::Receive ()

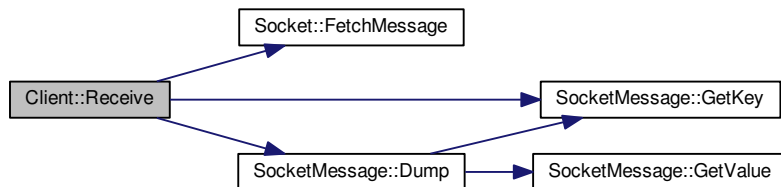
Receive a socket message from the master.

Here is the call graph for this function:



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

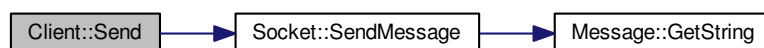
Here is the call graph for this function:



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

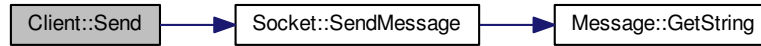
Send a message to the master through the socket.

Here is the call graph for this function:



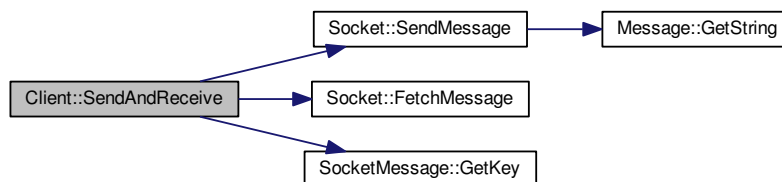
7.2.3.9 `void Client::Send (const Exception & e) const` `[inline]`

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:

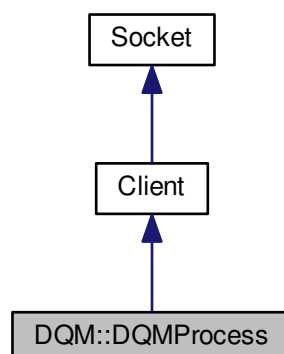
- include/QuarticCanvas.h

7.5 DQM::DQMProcess Class Reference

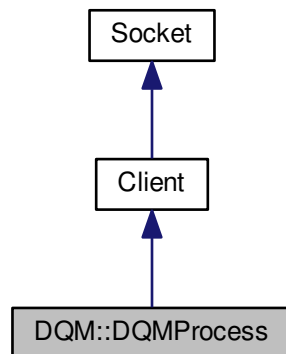
Handler for a common [DQM](#) process to run on the socket.

```
#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

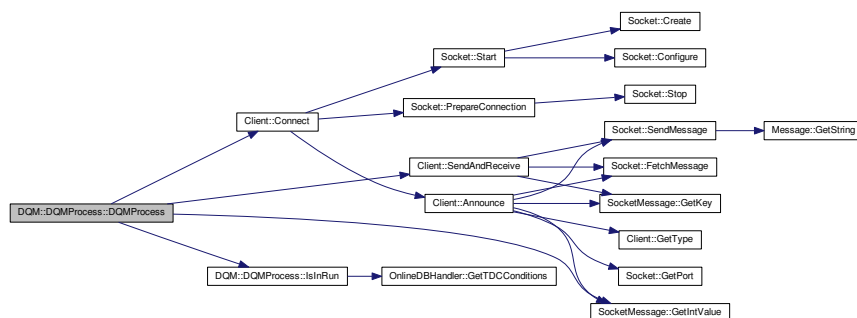
NewPlot

UpdatedPlot

7.5.3 Constructor & Destructor Documentation

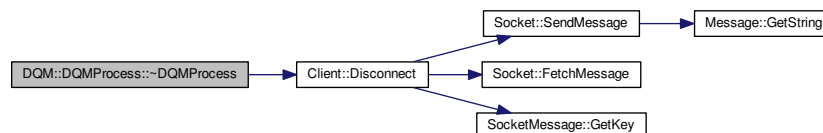
7.5.3.1 DQM::DQMProcess::DQMProcess (int *port*, unsigned short *order* = 0, const char * *det_type* = " ") [inline]

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

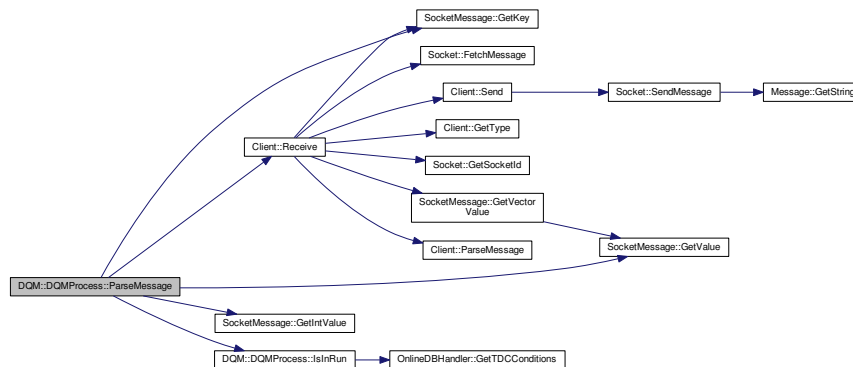
7.5.4.1 bool DQM::DQMProcess::IsInRun () [inline],[private]

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]

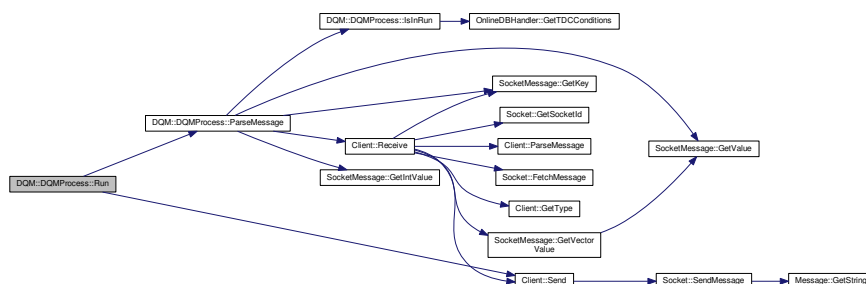
Here is the call graph for this function:



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.

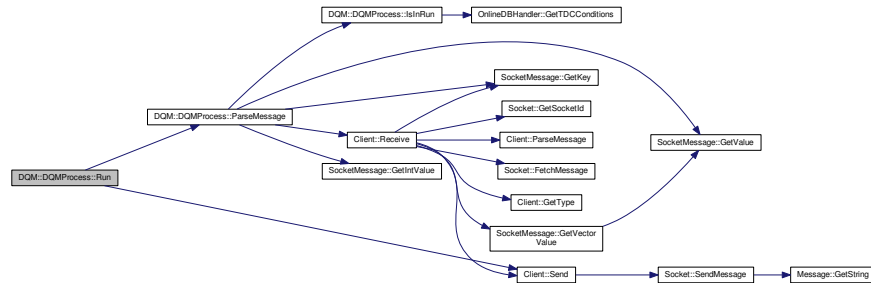
Here is the call graph for this function:



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:

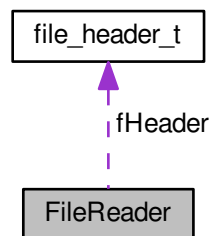
- `include/FileConstants.h`

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

<code>in</code>	<code>name</code>	Path to the file to read
<code>in</code>	<code>ro</code>	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

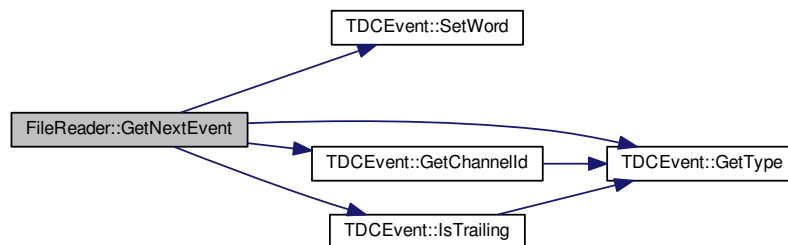
7.8.3.3 unsigned int FileReader::GetAcquisitionMode () const [inline]

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.

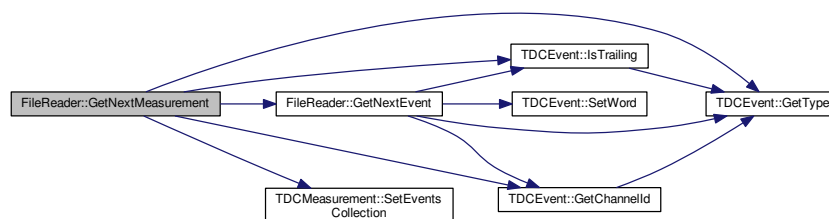
Parameters

in	<i>channel_id</i>	Unique identifier of the channel number to retrieve
out	<i>m</i>	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:

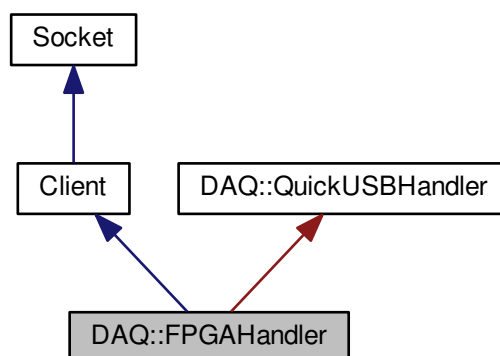
- include/FileReader.h
- src/FileReader.cpp

7.9 DAQ::FPGAHandler Class Reference

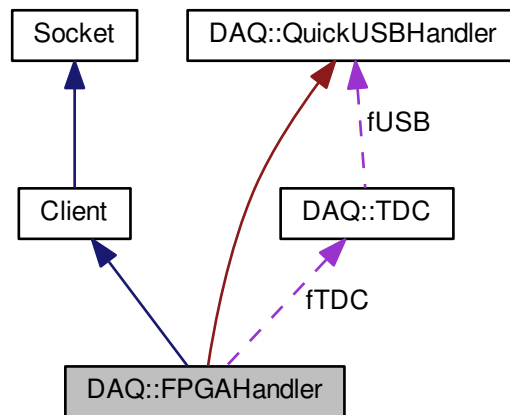
Driver for timing detectors' FPGA readout.

```
#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 flsFileOpen`
- `TDC * fTDC [NUM_HPTDC]`
- `bool flsTDCInReadout`
- `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

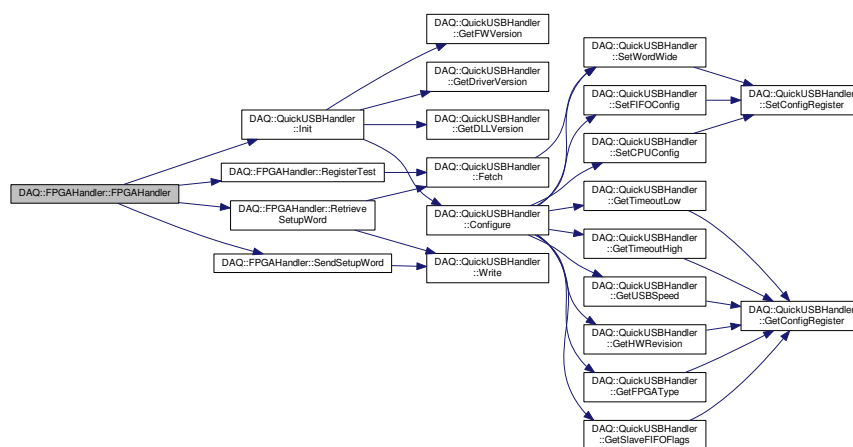
14 Apr 2015

7.9.2 Constructor & Destructor Documentation

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

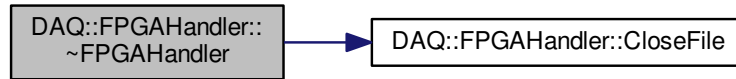
Bind to a FPGA through the USB protocol, and to the socket.

Here is the call graph for this function:



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 ()

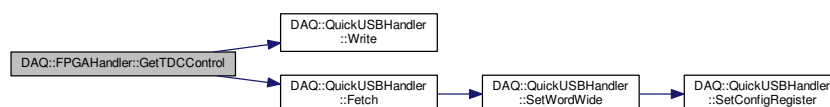
7.9.3.3 std::string DAQ::FPGAHandler::GetFilename () const [inline]

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::GetTDCCControl () const

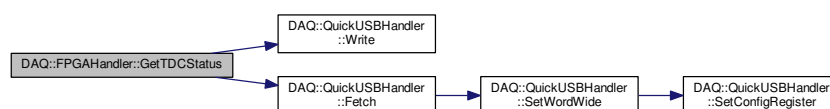
Here is the call graph for this function:



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:



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

[Socket](#) actor type retrieval method.

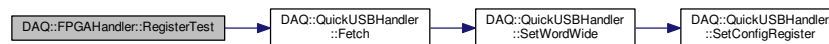
Reimplemented from [Client](#).

7.9.3.9 void DAQ::FPGAHandler::OpenFile ()

Open an output file to store header/HPTDC events.

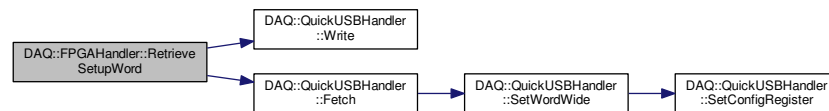
7.9.3.10 void DAQ::FPGAHandler::RegisterTest () const [private]

Here is the call graph for this function:



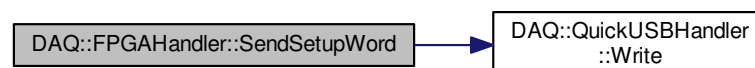
7.9.3.11 void DAQ::FPGAHandler::RetrieveSetupWord () [private]

Here is the call graph for this function:



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

Here is the call graph for this function:



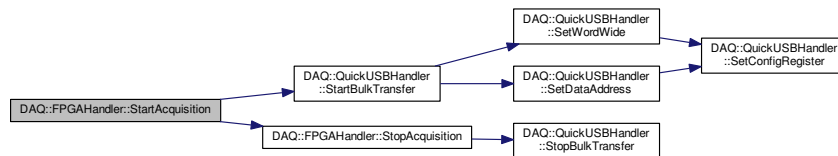
7.9.3.13 `void DAQ::FPGAHandler::SetTDCSetup (const TDCSetup & s) [inline]`

Here is the call graph for this function:



7.9.3.14 `void DAQ::FPGAHandler::StartAcquisition ()`

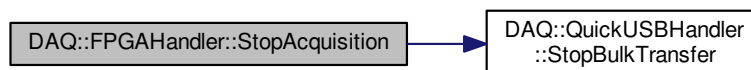
Here is the call graph for this function:



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:



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]

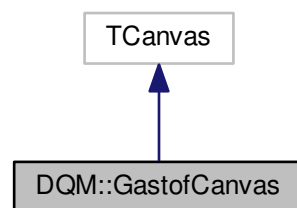
The documentation for this class was generated from the following files:

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

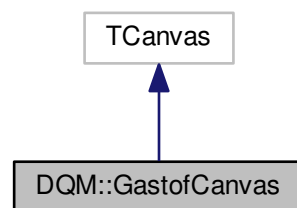
7.10 DQM::GastofCanvas Class Reference

```
#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

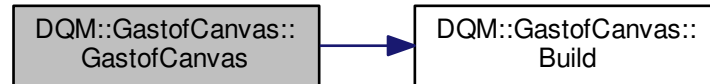
25 Jul 2015

7.10.2 Constructor & Destructor Documentation

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

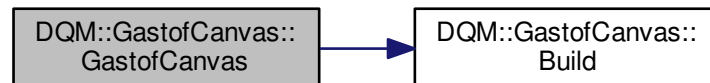
7.10.2.2 `DQM::GastofCanvas::GastofCanvas (TString name, unsigned int width = 500, unsigned int height = 500, TString upper_label = " ") [inline]`

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]`

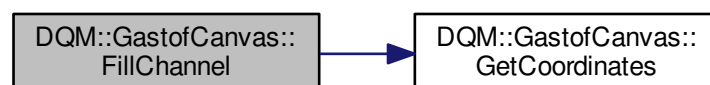
7.10.3 Member Function Documentation

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:

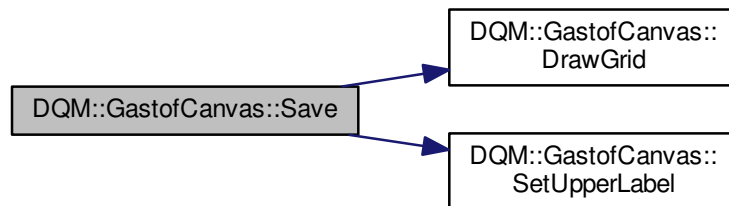


7.10.3.4 **Coord** DQM::GastofCanvas::GetCoordinates (unsigned short *nino_id*, unsigned short *channel_id*) const
[inline], [private]

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:

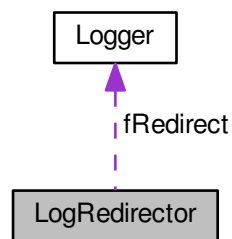
- `include/FileConstants.h`

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:

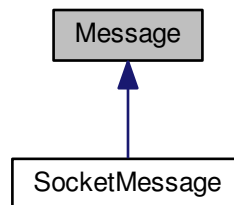
- `include/FileConstants.h`

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:

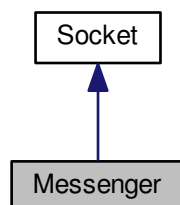
- include/Message.h

7.14 Messenger Class Reference

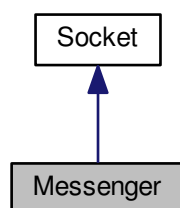
Base master object for the socket.

```
#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

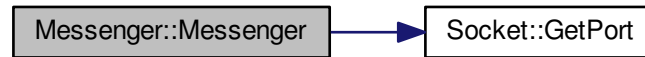
7.14.2.1 [Messenger::Messenger](#) ()

Build a void master object or socket actor.

7.14.2.2 Messenger::Messenger (int port)

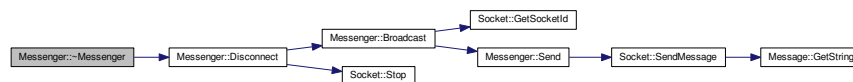
Build a master object to control the socket.

Here is the call graph for this function:



7.14.2.3 Messenger::~Messenger ()

Here is the call graph for this function:



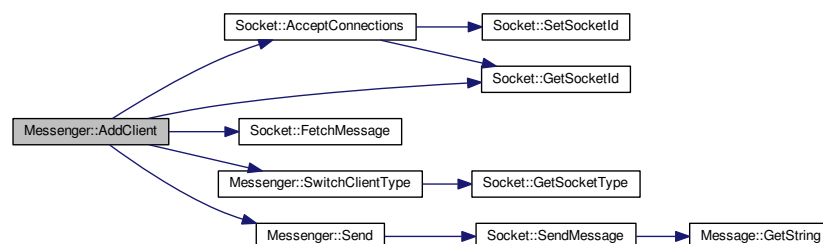
7.14.3 Member Function Documentation

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:



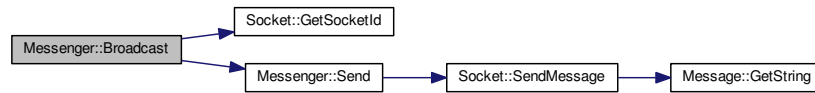
7.14.3.2 void Messenger::Broadcast (const Message & m) const

Emit a message to all clients connected through the socket.

Parameters

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

Here is the call graph for this function:

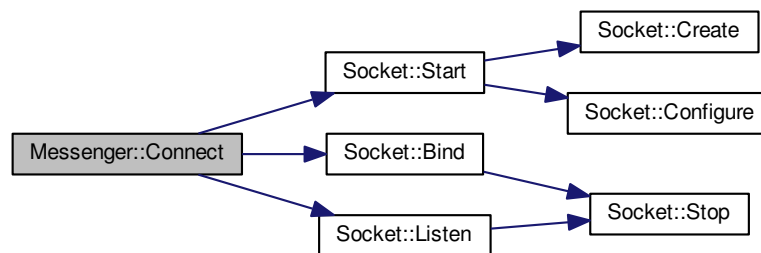


7.14.3.3 bool Messenger::Connect ()

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:

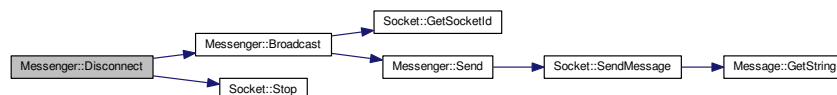


7.14.3.4 void Messenger::Disconnect ()

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]

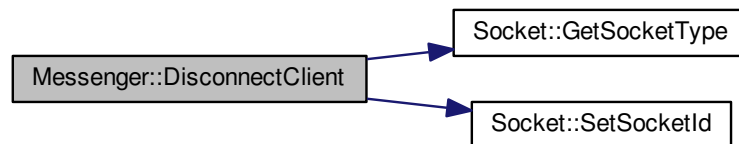
Disconnect a client.

Ask to a client to disconnect from this socket.

Parameters

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

Here is the call graph for this function:



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

Socket actor type retrieval method.

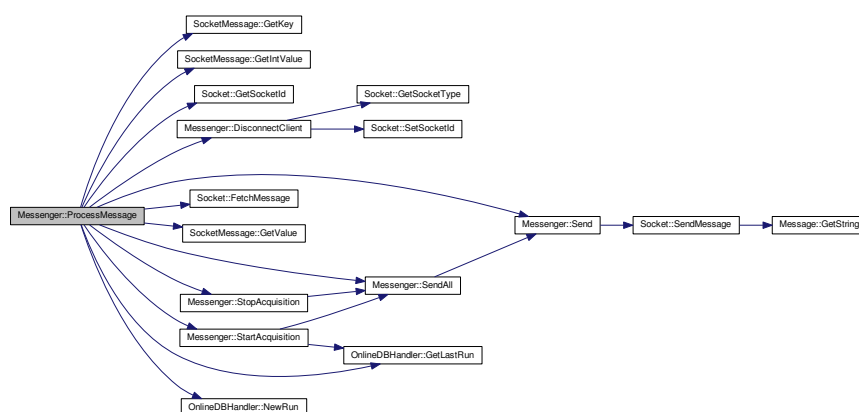
7.14.3.7 void Messenger::ProcessMessage (SocketMessage m, int sid) [private]

Process a message received from the socket.

Parameters

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

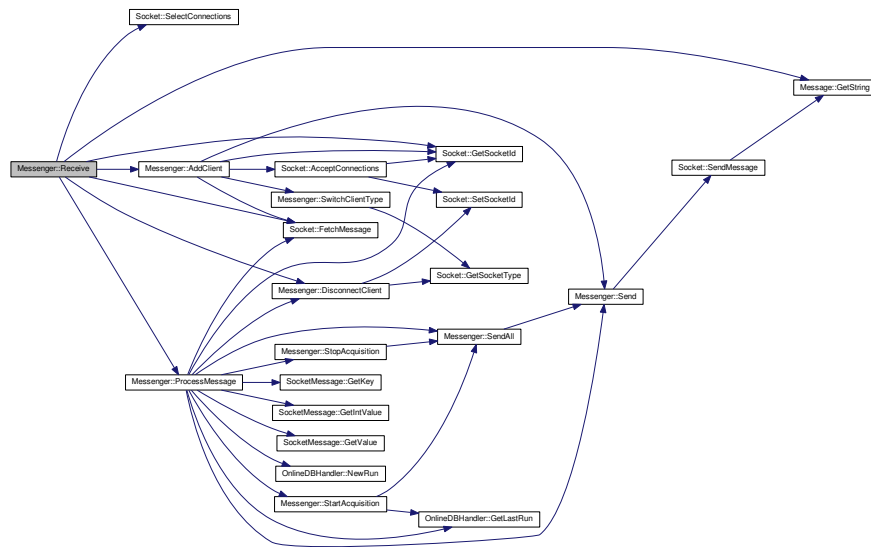
Here is the call graph for this function:



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

Here is the call graph for this function:



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	<i>type</i>	Client type
in	<i>m</i>	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]`

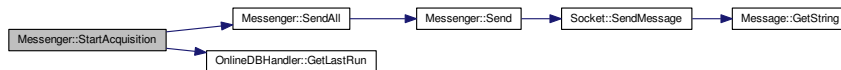
Here is the call graph for this function:



7.14.3.12 `void Messenger::StartAcquisition ()`

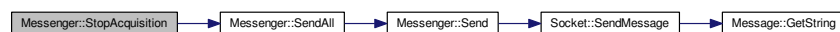
Start the data acquisition.

Here is the call graph for this function:



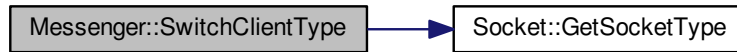
7.14.3.13 `void Messenger::StopAcquisition ()`

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 int 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:



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]`

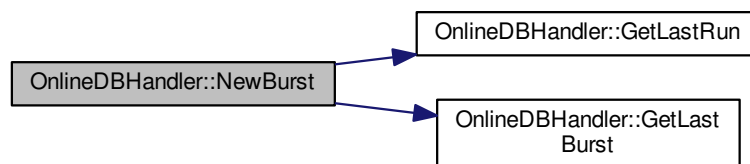
Retrieve information on a given run (spill IDs / timestamp)

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:



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 int 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:



7.15.5 Field Documentation

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

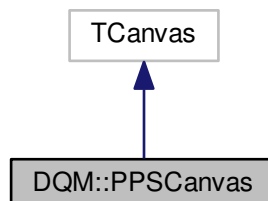
The documentation for this class was generated from the following file:

- `include/OnlineDBHandler.h`

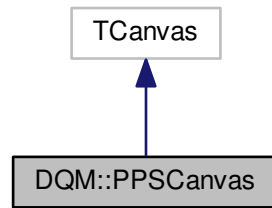
7.16 DQM::PPSCanvas Class Reference

```
#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

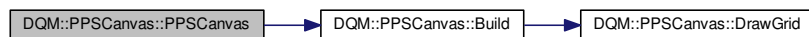
3 Aug 2015

7.16.2 Constructor & Destructor Documentation

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

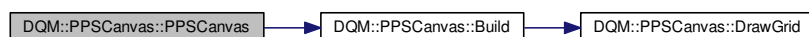
7.16.2.2 `DQM::PPSCanvas::PPSCanvas (TString name, unsigned int width = 500, unsigned int height = 500, TString upper_label = " ")` `[inline]`

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:

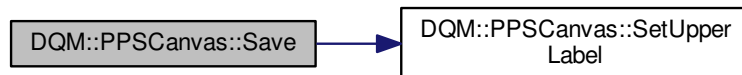


7.16.3.2 void DQM::PPSCanvas::DrawGrid () [inline],[private]

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]

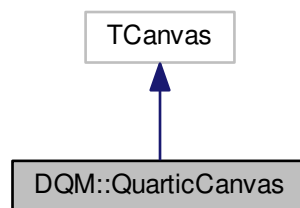
The documentation for this class was generated from the following file:

- include/PPSCanvas.h

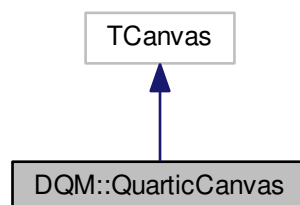
7.17 DQM::QuarticCanvas Class Reference

```
#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

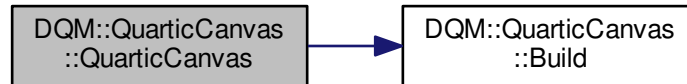
3 Aug 2015

7.17.2 Constructor & Destructor Documentation

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

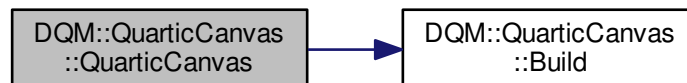
7.17.2.2 `DQM::QuarticCanvas::QuarticCanvas (TString name, unsigned int width = 500, unsigned int height = 500, TString upper_label = " ") [inline]`

Here is the call graph for this function:



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

Here is the call graph for this function:



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

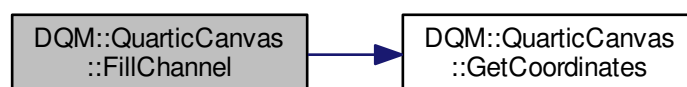
7.17.3 Member Function Documentation

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:

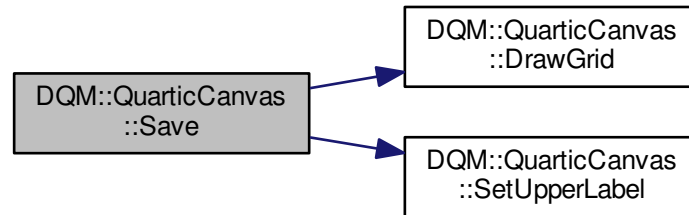


7.17.3.4 **Coord** DQM::QuarticCanvas::GetCoordinates (unsigned short *channel_id*) const [inline], [private]

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:



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:

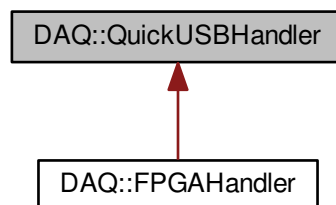
- `include/QuarticCanvas.h`

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 [flsStopping](#)

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,
 [kUSBFulSpeedForce](#) =0x0, [kUSBFulSpeedAllow](#) =0x8000 }
- enum [SPIConfig](#) {
 [kSPIENDIANlsb](#) =0x0, [kSPIENDIANmsb](#) =0x1, [kSPICPOLnormal](#) =0x0, [kSPICPOLinverted](#) =0x2,
 [kSPICPHAsampleclock](#) =0x0, [kSPICPHAclocksampling](#) =0x4, [kSPIPORTE](#) =0x0, [kSPIPORTA](#) =0x8,
 [kNCEPIN2](#) =0x0, [kNCEPIN7](#) =0x10, [kMISOPIN5](#) =0x0, [kMISOPIN2](#) =0x20,
 [kGPIFA8gpio](#) =0x0, [kGPIFA8gfadr8](#) =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 [SetSPICongig](#) (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
- [HWRRevision](#) [GetHWRRevision](#) () 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 [HWRRevision](#) &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
kl2CTL
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
kGPIFA8giadr8

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 ()

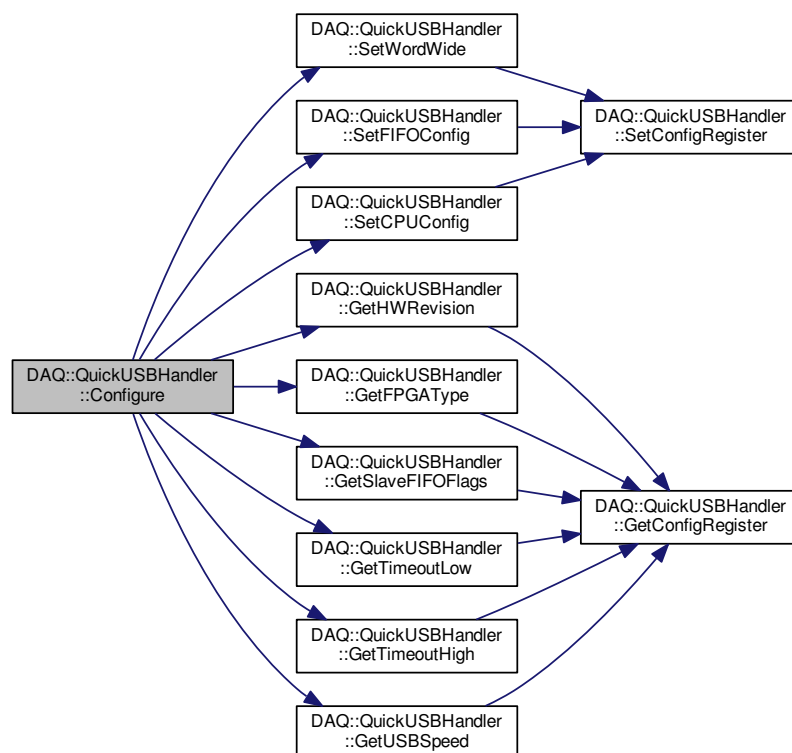
7.18.3.2 DAQ::QuickUSBHandler::~QuickUSBHandler () [virtual]

7.18.4 Member Function Documentation

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

Configure the board with the initial settings.

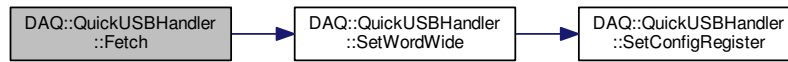
Here is the call graph for this function:



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:



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.

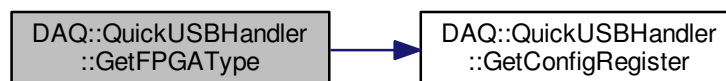
7.18.4.5 `QuickUSBHandler::Version DAQ::QuickUSBHandler::GetDriverVersion () const`

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:

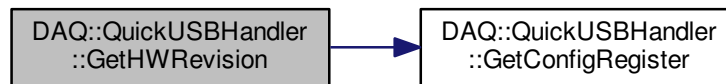


7.18.4.7 `QuickUSBHandler::Version DAQ::QuickUSBHandler::GetFWVersion () const`

Read the QuickUSB firmware revision.

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

Here is the call graph for this function:



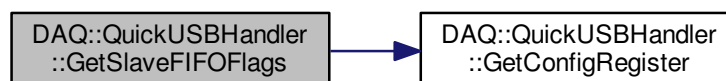
7.18.4.9 FIFOFlags DAQ::QuickUSBHandler::GetSlaveFIFOFlags () const [inline],[private]

Get the slave FIFO flag status

Note

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

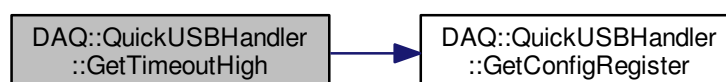
Here is the call graph for this function:



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:



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:



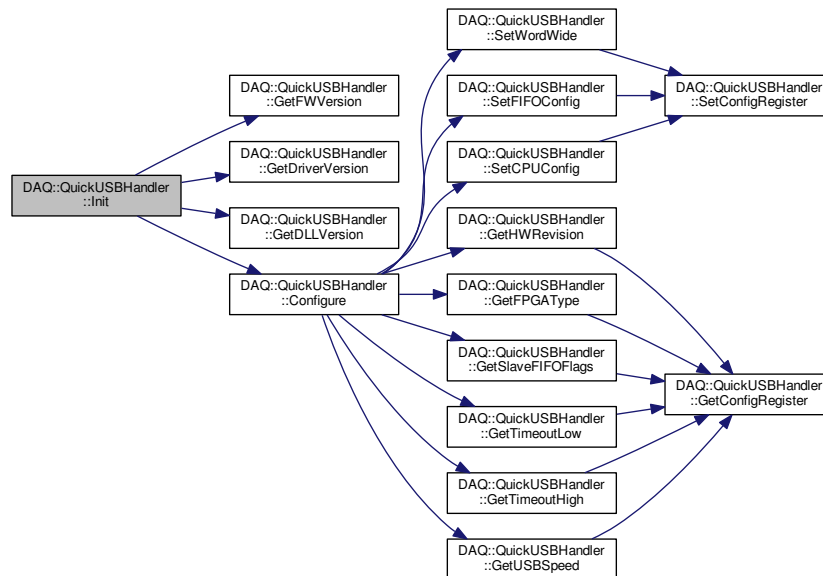
7.18.4.12 `USBSpeed DAQ::QuickUSBHandler::GetUSBSpeed () const [inline],[private]`

Here is the call graph for this function:



7.18.4.13 void DAQ::QuickUSBHandler::Init ()

Here is the call graph for this function:



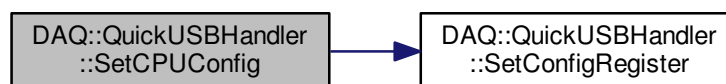
7.18.4.14 void DAQ::QuickUSBHandler::Reset () const

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:



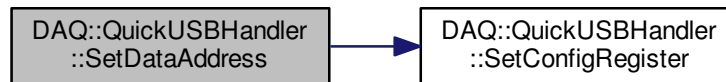
7.18.4.17 void DAQ::QuickUSBHandler::SetDataAddress (uint16_t addr, bool increment = false, bool enable_addr_bus = false) const [inline], [private]

Set the data bus starting address

Parameters

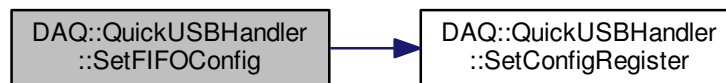
in	<i>increment</i>	Auto increment the bus address after each data transaction?
in	<i>enable_addr_↔ bus</i>	Enable the address bus?

Here is the call graph for this function:



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

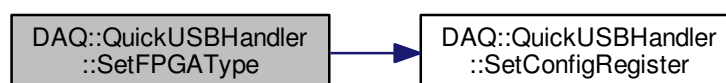
Here is the call graph for this function:



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

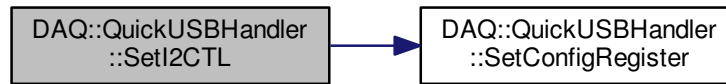
Set the FPGA configuration scheme.

Here is the call graph for this function:



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

Here is the call graph for this function:



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

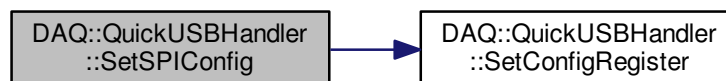
Here is the call graph for this function:



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

Configure the SPI interface.

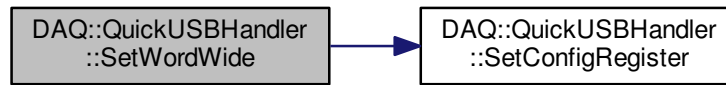
Here is the call graph for this function:



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:



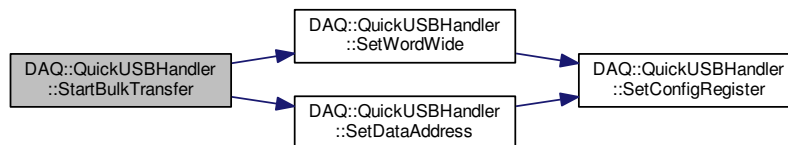
7.18.4.24 void DAQ::QuickUSBHandler::StartBulkTransfer (QVOIDRETURN callbackPQBULKSTREAM)

Start the data collection

Parameters

<code>in, out</code>	<code>callback</code>	Callback function called at the end of each data retrieval by a process
----------------------	-----------------------	-------------------------------------------------------------------------

Here is the call graph for this function:



7.18.4.25 void DAQ::QuickUSBHandler::StopBulkTransfer ()

Stop the data collection.

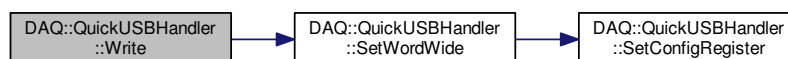
7.18.4.26 void DAQ::QuickUSBHandler::Write (uint16_t addr, uint8_t word) const [inline]

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:



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::flsStopping` [protected]

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

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

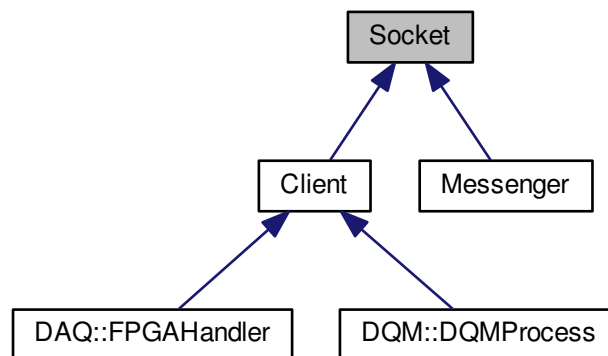
- `daq/include/QuickUSBHandler.h`
- `daq/src/QuickUSBHandler.cpp`

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)`

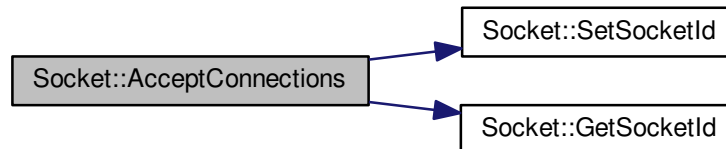
Accept connection from a client.

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

Parameters

<i>in, out</i>	<i>socket</i>	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.

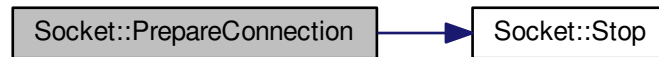
Set the socket to listen to any message coming from outside

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]

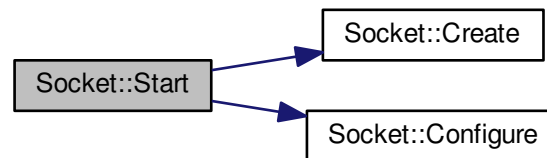
Start the socket.

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:

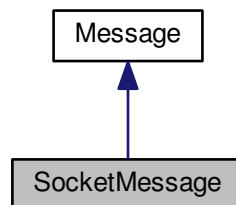
- include/Socket.h
- src/Socket.cpp

7.20 SocketMessage Class Reference

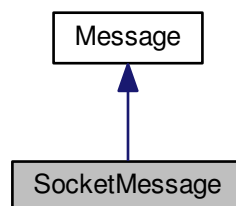
Socket-passed message type.

```
#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]

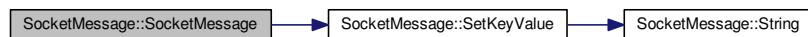
Here is the call graph for this function:



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

Construct a socket message out of a key.

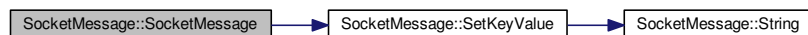
Here is the call graph for this function:



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.

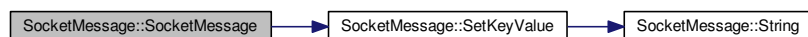
Here is the call graph for this function:



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.

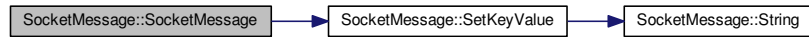
Here is the call graph for this function:



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.

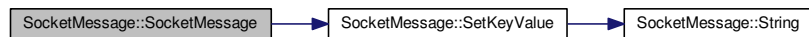
Here is the call graph for this function:



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.

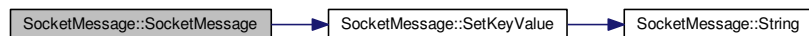
Here is the call graph for this function:



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.

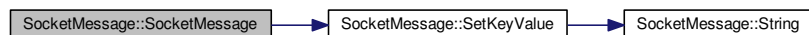
Here is the call graph for this function:



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.

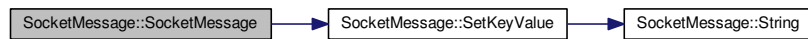
Here is the call graph for this function:



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]

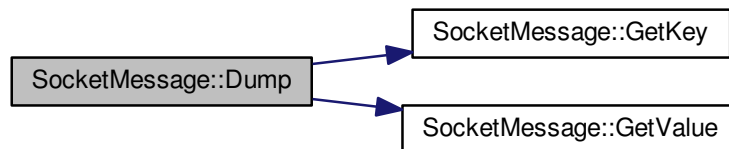
Construct a socket message out of a map of key/string-type value.

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.

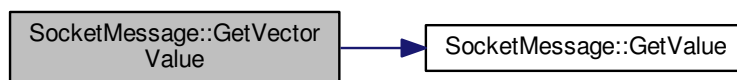
7.20.3.6 `std::string SocketMessage::GetValue () const` `[inline]`

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.

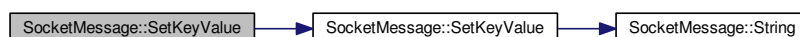
Here is the call graph for this function:



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

Send a short integer-valued message.

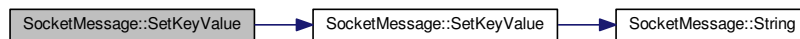
Here is the call graph for this function:



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

Send an integer-valued message.

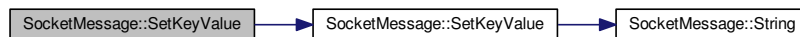
Here is the call graph for this function:



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

Send a long integer-valued message.

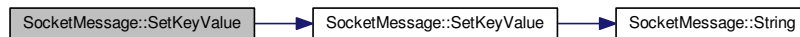
Here is the call graph for this function:



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:

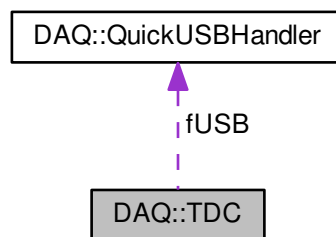
- include/SocketMessage.h

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, [OTRILING](#) = 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 [fld](#)
- [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
OTRILING
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::fld [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

TDMeasurement

TDCHheader

TDCTrailer

TDCErrror

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]

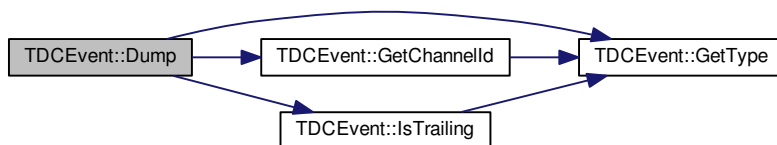
7.24.3.4 TDCEvent::TDCEvent (const EventType & ev) [inline]

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

<i>in</i>	<i>pair</i>	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:

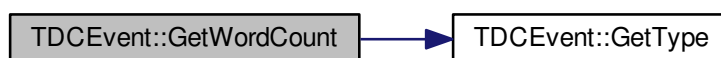


7.24.4.14 uint32_t TDCEvent::GetWord () const [inline]

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

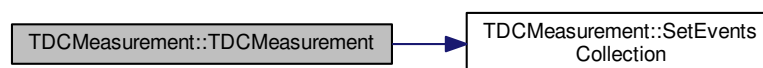
Jun 2015

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:

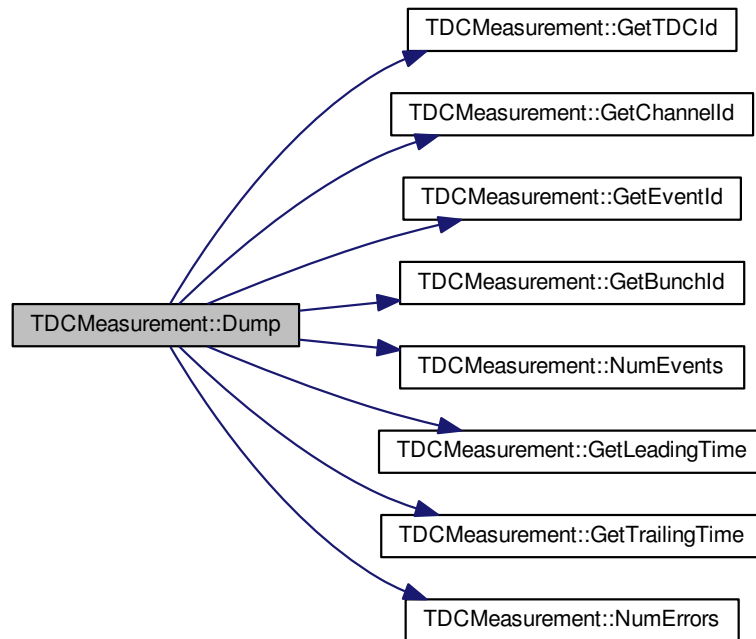


7.25.2.3 `TDCMeasurement::~~TDCMeasurement ()` `[inline]`

7.25.3 Member Function Documentation

7.25.3.1 void TDCMeasurement::Dump () [inline]

Here is the call graph for this function:



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]

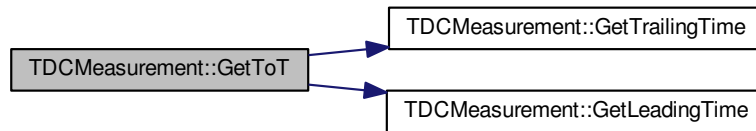
7.25.3.5 uint16_t TDCMeasurement::GetEventId () [inline]

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:



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

- ~Client
 - Client, [17](#)
- ~DQMProcess
 - DQM::DQMProcess, [24](#)
- ~FPGAHandler
 - DAQ::FPGAHandler, [33](#)
- ~FileReader
 - FileReader, [29](#)
- ~GastofCanvas
 - DQM::GastofCanvas, [39](#)
- ~Logger
 - Logger, [41](#)
- ~Message
 - Message, [44](#)
- ~Messenger
 - Messenger, [47](#)
- ~OnlineDBHandler
 - OnlineDBHandler, [54](#)
- ~PPSCanvas
 - DQM::PPSCanvas, [58](#)
- ~QuarticCanvas
 - DQM::QuarticCanvas, [62](#)
- ~QuickUSBHandler
 - DAQ::QuickUSBHandler, [69](#)
- ~Socket
 - Socket, [79](#)
- ~SocketMessage
 - SocketMessage, [89](#)
- ~TDC
 - DAQ::TDC, [94](#)
- ~TDCErrorFlag
 - TDCErrorFlag, [96](#)
- ~TDCEvent
 - TDCEvent, [99](#)
- ~TDCMeasurement
 - TDCMeasurement, [105](#)
- AcceptConnections
 - Socket, [79](#)
- acq_mode
 - file_header_t, [28](#)
- AcquisitionMode
 - HPTDC chip control, [11](#)
- Action
 - DQM::DQMProcess, [24](#)
- AddClient
 - Messenger, [47](#)
- AlterPassiveSerial
 - DAQ::QuickUSBHandler, [67](#)
- Announce

- Client, [18](#)
- Bind
 - Socket, [80](#)
- Broadcast
 - Messenger, [47](#)
- Build
 - DQM::GastofCanvas, [39](#)
 - DQM::PPSCanvas, [58](#)
 - DQM::QuarticCanvas, [62](#)
- BuildTables
 - OnlineDBHandler, [55](#)
- BuildVersion
 - DAQ::QuickUSBHandler::Version, [107](#)
- burst_id
 - OnlineDBHandler::BurstInfo, [15](#)
- BurstInfos
 - OnlineDBHandler, [54](#)
- c1
 - DQM::GastofCanvas, [40](#)
 - DQM::PPSCanvas, [59](#)
 - DQM::QuarticCanvas, [63](#)
- c2
 - DQM::GastofCanvas, [40](#)
 - DQM::PPSCanvas, [59](#)
 - DQM::QuarticCanvas, [63](#)
- CLIENT
 - Socket, [79](#)
- CONT_STORAGE
 - HPTDC chip control, [11](#)
- CPUConfig
 - DAQ::QuickUSBHandler, [67](#)
- CY7C68013A
 - DAQ::QuickUSBHandler, [67](#)
- CY7C68013AB
 - DAQ::QuickUSBHandler, [67](#)
- CY7C68013CD
 - DAQ::QuickUSBHandler, [67](#)
- CY7C68013E
 - DAQ::QuickUSBHandler, [67](#)
- CheckFirmwareVersion
 - DAQ::TDC, [94](#)
- Clear
 - FileReader, [30](#)
- Client, [15](#)
 - ~Client, [17](#)
 - Announce, [18](#)
 - Client, [17](#)
 - Connect, [18](#)

- Disconnect, [19](#)
- fClientId, [21](#)
- flsConnected, [21](#)
- fType, [21](#)
- GetType, [19](#)
- ParseMessage, [19](#)
- Receive, [19](#), [20](#)
- Send, [20](#)
- SendAndReceive, [21](#)
- CloseFile
 - DAQ::FPGAHandler, [34](#)
- Configure
 - DAQ::QuickUSBHandler, [69](#)
 - Socket, [80](#)
- Connect
 - Client, [18](#)
 - Messenger, [48](#)
- contents
 - LogRedirector, [42](#)
- Create
 - Socket, [80](#)
- DAQ, [13](#)
 - operator<<, [13](#)
 - Socket, [79](#)
- DAQ::FPGAHandler, [31](#)
 - ~FPGAHandler, [33](#)
 - CloseFile, [34](#)
 - ErrorState, [34](#)
 - fFilename, [36](#)
 - flsFileOpen, [36](#)
 - flsTDCInReadout, [36](#)
 - fOutput, [36](#)
 - FPGAHandler, [33](#)
 - fSetupReg, [36](#)
 - fTDC, [36](#)
 - GetFilename, [34](#)
 - GetTDC, [34](#)
 - GetTDCControl, [34](#)
 - GetTDCSetup, [34](#)
 - GetTDCStatus, [34](#)
 - GetType, [35](#)
 - OpenFile, [35](#)
 - RegisterTest, [35](#)
 - RetrieveSetupWord, [35](#)
 - SendSetupWord, [35](#)
 - SetTDCSetup, [35](#)
 - StartAcquisition, [36](#)
 - Stop, [36](#)
 - StopAcquisition, [36](#)
- DAQ::QuickUSBHandler, [64](#)
 - ~QuickUSBHandler, [69](#)
 - AlteraPassiveSerial, [67](#)
 - CPUConfig, [67](#)
 - CY7C68013A, [67](#)
 - CY7C68013AB, [67](#)
 - CY7C68013CD, [67](#)
 - CY7C68013E, [67](#)
 - Configure, [69](#)
 - fDevice, [77](#)
 - fHandle, [77](#)
 - flsStopping, [77](#)
 - FPGAType, [67](#)
 - fStreamId, [77](#)
 - Fetch, [69](#)
 - GetConfigRegister, [70](#)
 - GetDLLVersion, [70](#)
 - GetDriverVersion, [70](#)
 - GetFPGAType, [70](#)
 - GetFWVersion, [70](#)
 - GetHWRevision, [70](#)
 - GetSlaveFIFOFlags, [71](#)
 - GetTimeoutHigh, [71](#)
 - GetTimeoutLow, [71](#)
 - GetUSBSpeed, [72](#)
 - HWRevision, [67](#)
 - I2CTL, [67](#)
 - Init, [72](#)
 - k16bits, [69](#)
 - k8bits, [69](#)
 - kBusError, [67](#)
 - kCLKINVdisable, [67](#)
 - kCLKINVenable, [67](#)
 - kCLKOUTdisable, [67](#)
 - kCLKOUTenable, [67](#)
 - kCLKSPD12MHz, [67](#)
 - kCLKSPD24MHz, [67](#)
 - kCLKSPD48MHz, [67](#)
 - kCLKSPDreserved, [67](#)
 - kCPUConfig, [68](#)
 - kDataAddress, [68](#)
 - kFIFOConfig, [68](#)
 - kFPGAType, [68](#)
 - kGPIFA8gfiadr8, [68](#)
 - kGPIFA8gpio, [68](#)
 - kHandleACK, [67](#)
 - kHighLogic, [68](#)
 - kl2CBusClkSpeed100kHz, [67](#)
 - kl2CBusClkSpeed400kHz, [67](#)
 - kl2CTL, [68](#)
 - kIgnoreACK, [67](#)
 - kLowLogic, [68](#)
 - kMISOPIN2, [68](#)
 - kMISOPIN5, [68](#)
 - kNCEPIN2, [68](#)
 - kNCEPIN7, [68](#)
 - kNoACK, [67](#)
 - kNormalCompletion, [67](#)
 - kPinFlags, [68](#)
 - kPortA, [68](#)
 - kPortAConfig, [68](#)
 - kPortB, [68](#)
 - kPortC, [68](#)
 - kPortD, [68](#)
 - kPortE, [68](#)
 - kSPICPHAclocksample, [68](#)
 - kSPICPHAsampleclock, [68](#)

- kSPICPOLinverted, 68
- kSPICPOLnormal, 68
- kSPIConfig, 68
- kSPIENDIANIsb, 68
- kSPIENDIANmsb, 68
- kSPIPORTA, 68
- kSPIPORTE, 68
- kSlaveFIFOFlags, 68
- kSlaveWait, 67
- kTimeout, 67
- kTimeoutHigh, 68
- kTimeoutLow, 68
- kUSBFullSpeedAllow, 67
- kUSBFullSpeedForce, 67
- kVersionSpeed, 68
- kWordWide, 68
- LogicLevel, 67
- operator<<, 77
- QuickUSBHandler, 69
- Reset, 73
- SPIConfig, 68
- SetCPUConfig, 73
- SetConfigRegister, 73
- SetDataAddress, 73
- SetFIFOConfig, 74
- SetFPGAType, 74
- SetI2CTL, 74
- SetPort, 75
- SetSPIConfig, 75
- SetWordWide, 75
- SettingsRegister, 68
- StartBulkTransfer, 76
- StopBulkTransfer, 76
- USBFullSpeed, 69
- USBHighSpeed, 69
- USBSpeed, 68
- WordWide, 69
- Write, 76
- XilinxSlaveSerial, 67
- DAQ::QuickUSBHandler::FIFOFlags, 26
- operator<<, 27
- RDY0, 27
- RDY1, 27
- ReadFIFOEmpty, 27
- ReadFIFOFull, 27
- WriteFIFOEmpty, 27
- WriteFIFOFull, 27
- DAQ::QuickUSBHandler::Version, 107
- BuildVersion, 107
- MajorVersion, 107
- MinorVersion, 107
- DAQ::TDC, 92
- ~TDC, 94
- CheckFirmwareVersion, 94
- DetectionMode, 93
- fBS, 94
- fControl, 94
- fId, 94
- fSetup, 94
- fStatus, 94
- fUSB, 94
- FetchEvents, 94
- GetSetupRegister, 94
- OLEADING, 93
- OTRILING, 93
- PAIR, 93
- ReadConfiguration, 94
- ReadRegister, 94
- ReadStatus, 94
- SendConfiguration, 94
- SetSetupRegister, 94
- SoftReset, 94
- TDC, 93
- TRAILEAD, 93
- WriteRegister, 94
- DETECTOR
- Socket, 79
- DQM, 13
- Socket, 79
- DQM::DQMProcess, 22
- ~DQMProcess, 24
- Action, 24
- DQMProcess, 24
- fAddressesCanProcess, 26
- fDetectorType, 26
- fOrder, 26
- fRunNumber, 26
- IsInRun, 24
- NewPlot, 24
- ParseMessage, 25
- Run, 25
- UpdatedPlot, 24
- DQM::GastofCanvas, 37
- ~GastofCanvas, 39
- Build, 39
- c1, 40
- c2, 40
- DrawGrid, 39
- fBoardId, 40
- fHeight, 40
- fHist, 40
- fLabel1, 40
- fLabel2, 40
- fLabel3, 40
- fLabel4, 40
- fLabelsDrawn, 40
- fLegend, 40
- fLegendNumEntries, 40
- fLegendX, 40
- fLegendY, 41
- fRunDate, 41
- fRunId, 41
- fSpillId, 41
- fUpperLabel, 41
- fUpperLabelText, 41
- fWidth, 41

- FillChannel, 39
- GastofCanvas, 38, 39
- GetCoordinates, 40
- Grid, 40
- Save, 40
- SetRunInfo, 40
- SetUpperLabel, 40
- DQM::GastofCanvas::Coord, 21
 - x, 21
 - y, 22
- DQM::PPSCanvas, 56
 - ~PPSCanvas, 58
 - Build, 58
 - c1, 59
 - c2, 59
 - DrawGrid, 58
 - fHeight, 59
 - fLabel1, 59
 - fLabel2, 59
 - fLabel3, 59
 - fLabelsDrawn, 59
 - fLegend, 59
 - fLegendNumEntries, 59
 - fLegendX, 59
 - fLegendY, 59
 - fRunDate, 59
 - fRunId, 59
 - fUpperLabel, 59
 - fUpperLabelText, 59
 - fWidth, 59
 - Grid, 59
 - PPSCanvas, 58
 - Save, 59
 - SetRunInfo, 59
 - SetUpperLabel, 59
- DQM::QuarticCanvas, 60
 - ~QuarticCanvas, 62
 - Build, 62
 - c1, 63
 - c2, 63
 - DrawGrid, 62
 - fBoardId, 63
 - fHeight, 63
 - fHist, 63
 - fLabel1, 63
 - fLabel2, 63
 - fLabel3, 63
 - fLabel4, 63
 - fLabelsDrawn, 63
 - fLegend, 63
 - fLegendNumEntries, 63
 - fLegendX, 63
 - fLegendY, 64
 - fRunDate, 64
 - fRunId, 64
 - fSpillId, 64
 - fUpperLabel, 64
 - fUpperLabelText, 64
 - fWidth, 64
 - FillChannel, 62
 - GetCoordinates, 62
 - Grid, 63
 - QuarticCanvas, 61, 62
 - Save, 63
 - SetRunInfo, 63
 - SetUpperLabel, 63
- DQM::QuarticCanvas::Coord, 22
 - x, 22
 - y, 22
- DQMProcess
 - DQM::DQMProcess, 24
- det_mode
 - file_header_t, 28
- DetectionMode
 - DAQ::TDC, 93
- detector
 - OnlineDBHandler::TDCConditions, 95
- Disconnect
 - Client, 19
 - Messenger, 48
- DisconnectClient
 - Messenger, 48
- DrawGrid
 - DQM::GastofCanvas, 39
 - DQM::PPSCanvas, 58
 - DQM::QuarticCanvas, 62
- Dump
 - FileReader, 30
 - Message, 44
 - SocketMessage, 89
 - TDCErrorFlag, 96
 - TDCEvent, 99
 - TDCMeasurement, 105
- DumpConnected
 - Socket, 80
- ETTT
 - TDCEvent, 99
- ErrorState
 - DAQ::FPGAHandler, 34
- EventType
 - TDCEvent, 99
- fAddress
 - Socket, 83
- fAddressesCanProcess
 - DQM::DQMProcess, 26
- fBS
 - DAQ::TDC, 94
- fBoardId
 - DQM::GastofCanvas, 40
 - DQM::QuarticCanvas, 63
- fBuffer
 - Logger, 41
 - Socket, 83
- fClientId
 - Client, 21

- fControl
 - DAQ::TDC, [94](#)
- fDB
 - OnlineDBHandler, [56](#)
- fDetectorType
 - DQM::DQMProcess, [26](#)
- fDevice
 - DAQ::QuickUSBHandler, [77](#)
- fEvents
 - TDCMeasurement, [107](#)
- fFile
 - FileReader, [31](#)
- fFilename
 - DAQ::FPGAHandler, [36](#)
- fHandle
 - DAQ::QuickUSBHandler, [77](#)
- fHeader
 - FileReader, [31](#)
- fHeight
 - DQM::GastofCanvas, [40](#)
 - DQM::PPSCanvas, [59](#)
 - DQM::QuarticCanvas, [63](#)
- fHist
 - DQM::GastofCanvas, [40](#)
 - DQM::QuarticCanvas, [63](#)
- fId
 - DAQ::TDC, [94](#)
- fIsConnected
 - Client, [21](#)
- fIsFileOpen
 - DAQ::FPGAHandler, [36](#)
- fIsStopping
 - DAQ::QuickUSBHandler, [77](#)
- fIsTDCInReadout
 - DAQ::FPGAHandler, [36](#)
- fLabel1
 - DQM::GastofCanvas, [40](#)
 - DQM::PPSCanvas, [59](#)
 - DQM::QuarticCanvas, [63](#)
- fLabel2
 - DQM::GastofCanvas, [40](#)
 - DQM::PPSCanvas, [59](#)
 - DQM::QuarticCanvas, [63](#)
- fLabel3
 - DQM::GastofCanvas, [40](#)
 - DQM::PPSCanvas, [59](#)
 - DQM::QuarticCanvas, [63](#)
- fLabel4
 - DQM::GastofCanvas, [40](#)
 - DQM::QuarticCanvas, [63](#)
- fLabelsDrawn
 - DQM::GastofCanvas, [40](#)
 - DQM::PPSCanvas, [59](#)
 - DQM::QuarticCanvas, [63](#)
- fLegend
 - DQM::GastofCanvas, [40](#)
 - DQM::PPSCanvas, [59](#)
 - DQM::QuarticCanvas, [63](#)
- fLegendNumEntries
 - DQM::GastofCanvas, [40](#)
 - DQM::PPSCanvas, [59](#)
 - DQM::QuarticCanvas, [63](#)
- fLegendX
 - DQM::GastofCanvas, [40](#)
 - DQM::PPSCanvas, [59](#)
 - DQM::QuarticCanvas, [63](#)
- fLegendY
 - DQM::GastofCanvas, [41](#)
 - DQM::PPSCanvas, [59](#)
 - DQM::QuarticCanvas, [64](#)
- fMap
 - TDCMeasurement, [107](#)
- fMaster
 - Socket, [83](#)
- fMessage
 - SocketMessage, [91](#)
- fNumAttempts
 - Messenger, [53](#)
- fNumEvents
 - FileReader, [31](#)
- fOrder
 - DQM::DQMProcess, [26](#)
- fOutput
 - DAQ::FPGAHandler, [36](#)
- FPGA board control, [10](#)
- FPGAHandler
 - DAQ::FPGAHandler, [33](#)
- FPGAType
 - DAQ::QuickUSBHandler, [67](#)
- fPID
 - Messenger, [53](#)
- fPort
 - Socket, [83](#)
- fReadFds
 - Socket, [83](#)
- fReadoutMode
 - FileReader, [31](#)
- fRedirect
 - LogRedirector, [43](#)
- fRunDate
 - DQM::GastofCanvas, [41](#)
 - DQM::PPSCanvas, [59](#)
 - DQM::QuarticCanvas, [64](#)
- fRunId
 - DQM::GastofCanvas, [41](#)
 - DQM::PPSCanvas, [59](#)
 - DQM::QuarticCanvas, [64](#)
- fRunNumber
 - DQM::DQMProcess, [26](#)
- fSS
 - LogRedirector, [43](#)
- fSetup
 - DAQ::TDC, [94](#)
- fSetupReg
 - DAQ::FPGAHandler, [36](#)
- fSocketId

- Socket, [83](#)
- fSocketsConnected
 - Socket, [83](#)
- fSpillId
 - DQM::GastofCanvas, [41](#)
 - DQM::QuarticCanvas, [64](#)
- fStatus
 - DAQ::TDC, [94](#)
- fStderrPipe
 - Messenger, [53](#)
- fStdoutPipe
 - Messenger, [53](#)
- fStream
 - Logger, [41](#)
- fStreamId
 - DAQ::QuickUSBHandler, [77](#)
- fString
 - Message, [44](#)
- fTDC
 - DAQ::FPGAHandler, [36](#)
- fType
 - Client, [21](#)
- fUSB
 - DAQ::TDC, [94](#)
- fUpperLabel
 - DQM::GastofCanvas, [41](#)
 - DQM::PPSCanvas, [59](#)
 - DQM::QuarticCanvas, [64](#)
- fUpperLabelText
 - DQM::GastofCanvas, [41](#)
 - DQM::PPSCanvas, [59](#)
 - DQM::QuarticCanvas, [64](#)
- fWidth
 - DQM::GastofCanvas, [41](#)
 - DQM::PPSCanvas, [59](#)
 - DQM::QuarticCanvas, [64](#)
- fWord
 - TDCErrFlag, [97](#)
 - TDCEvent, [104](#)
- fWriteTime
 - FileReader, [31](#)
- Fetch
 - DAQ::QuickUSBHandler, [69](#)
- FetchEvents
 - DAQ::TDC, [94](#)
- FetchMessage
 - Socket, [80](#)
- file_header_t, [27](#)
 - acq_mode, [28](#)
 - det_mode, [28](#)
 - magic, [28](#)
 - num_hptdc, [28](#)
 - run_id, [28](#)
 - spill_id, [28](#)
- FileReader, [28](#)
 - ~FileReader, [29](#)
 - Clear, [30](#)
 - Dump, [30](#)
 - fFile, [31](#)
 - fHeader, [31](#)
 - fNumEvents, [31](#)
 - fReadoutMode, [31](#)
 - fWriteTime, [31](#)
 - FileReader, [29](#)
 - GetAcquisitionMode, [30](#)
 - GetBurstId, [30](#)
 - GetDetectionMode, [30](#)
 - GetNextEvent, [30](#)
 - GetNextMeasurement, [30](#)
 - GetNumEvents, [31](#)
 - GetNumTDCs, [31](#)
 - GetRunId, [31](#)
 - IsOpen, [31](#)
 - Open, [31](#)
- FillChannel
 - DQM::GastofCanvas, [39](#)
 - DQM::QuarticCanvas, [62](#)
- Filler
 - TDCEvent, [99](#)
- GastofCanvas
 - DQM::GastofCanvas, [38](#), [39](#)
- GetAcquisitionMode
 - FileReader, [30](#)
- GetBunchId
 - TDCEvent, [99](#)
 - TDCMeasurement, [106](#)
- GetBurstId
 - FileReader, [30](#)
- GetChannelId
 - TDCEvent, [100](#)
 - TDCMeasurement, [106](#)
- GetCleanedValue
 - SocketMessage, [89](#)
- GetConfigRegister
 - DAQ::QuickUSBHandler, [70](#)
- GetCoordinates
 - DQM::GastofCanvas, [40](#)
 - DQM::QuarticCanvas, [62](#)
- GetDLLVersion
 - DAQ::QuickUSBHandler, [70](#)
- GetDetectionMode
 - FileReader, [30](#)
- GetDriverVersion
 - DAQ::QuickUSBHandler, [70](#)
- GetETTT
 - TDCEvent, [100](#)
 - TDCMeasurement, [106](#)
- GetErrorFlags
 - TDCEvent, [100](#)
- GetEventCount
 - TDCEvent, [101](#)
- GetEventId
 - TDCEvent, [101](#)
 - TDCMeasurement, [106](#)
- GetFPGAType
 - DAQ::QuickUSBHandler, [70](#)

- GetFWVersion
 - DAQ::QuickUSBHandler, 70
- GetFilename
 - DAQ::FPGAHandler, 34
- GetGeo
 - TDCEvent, 101
- GetHWRevision
 - DAQ::QuickUSBHandler, 70
- GetIntValue
 - SocketMessage, 89
- GetKey
 - Message, 44
 - SocketMessage, 89
- GetLastBurst
 - OnlineDBHandler, 55
- GetLastRun
 - OnlineDBHandler, 55
- GetLeadingTime
 - TDCMeasurement, 106
- GetNextEvent
 - FileReader, 30
- GetNextMeasurement
 - FileReader, 30
- GetNumEvents
 - FileReader, 31
- GetNumTDCs
 - FileReader, 31
- GetPort
 - Socket, 81
- GetRunId
 - FileReader, 31
- GetRunInfo
 - OnlineDBHandler, 55
- GetRuns
 - OnlineDBHandler, 55
- GetSetupRegister
 - DAQ::TDC, 94
- GetSlaveFIFOFlags
 - DAQ::QuickUSBHandler, 71
- GetSocketId
 - Socket, 81
- GetSocketType
 - Socket, 81
- GetStatus
 - TDCEvent, 102
- GetString
 - Message, 44
 - SocketMessage, 89
- GetTDC
 - DAQ::FPGAHandler, 34
- GetTDCConditions
 - OnlineDBHandler, 55
- GetTDCControl
 - DAQ::FPGAHandler, 34
- GetTDCId
 - TDCEvent, 102
 - TDCMeasurement, 106
- GetTDCSetup
 - DAQ::FPGAHandler, 34
- GetTDCStatus
 - DAQ::FPGAHandler, 34
- GetTime
 - TDCEvent, 102
- GetTimeoutHigh
 - DAQ::QuickUSBHandler, 71
- GetTimeoutLow
 - DAQ::QuickUSBHandler, 71
- GetToT
 - TDCMeasurement, 106
- GetTrailingTime
 - TDCMeasurement, 107
- GetType
 - Client, 19
 - DAQ::FPGAHandler, 35
 - Messenger, 50
 - TDCEvent, 103
- GetUSBSpeed
 - DAQ::QuickUSBHandler, 72
- GetValue
 - SocketMessage, 89
- GetVectorValue
 - SocketMessage, 90
- GetWidth
 - TDCEvent, 103
- GetWord
 - TDCErrorFlag, 96
 - TDCEvent, 103
- GetWordCount
 - TDCEvent, 103
- GlobalHeader
 - TDCEvent, 99
- GlobalTrailer
 - TDCEvent, 99
- Grid
 - DQM::GastofCanvas, 40
 - DQM::PPSCanvas, 59
 - DQM::QuarticCanvas, 63
- HPTDC chip control, 11
 - AcquisitionMode, 11
 - CONT_STORAGE, 11
 - TRIG_MATCH, 11
- HWRevision
 - DAQ::QuickUSBHandler, 67
- HasGroupError
 - TDCErrorFlag, 96
- HasInternalChipError
 - TDCErrorFlag, 96
- HasL1BufferOverflow
 - TDCErrorFlag, 97
- HasReachedEventSizeLimit
 - TDCErrorFlag, 97
- HasReadoutFIFOOverflow
 - TDCErrorFlag, 97
- HasTriggerFIFOOverflow
 - TDCErrorFlag, 97

- I2CTL
 - DAQ::QuickUSBHandler, [67](#)
- INVALID
 - Socket, [79](#)
- Init
 - DAQ::QuickUSBHandler, [72](#)
- IsFromWeb
 - Message, [44](#)
- IsInRun
 - DQM::DQMProcess, [24](#)
- IsOpen
 - FileReader, [31](#)
- IsTrailing
 - TDCEvent, [103](#)
- IsWebSocket
 - Socket, [81](#)
- k16bits
 - DAQ::QuickUSBHandler, [69](#)
- k8bits
 - DAQ::QuickUSBHandler, [69](#)
- kBusError
 - DAQ::QuickUSBHandler, [67](#)
- kCLKINVdisable
 - DAQ::QuickUSBHandler, [67](#)
- kCLKINVenable
 - DAQ::QuickUSBHandler, [67](#)
- kCLKOUTdisable
 - DAQ::QuickUSBHandler, [67](#)
- kCLKOUTenable
 - DAQ::QuickUSBHandler, [67](#)
- kCLKSPD12MHz
 - DAQ::QuickUSBHandler, [67](#)
- kCLKSPD24MHz
 - DAQ::QuickUSBHandler, [67](#)
- kCLKSPD48MHz
 - DAQ::QuickUSBHandler, [67](#)
- kCLKSPDreserved
 - DAQ::QuickUSBHandler, [67](#)
- kCPUConfig
 - DAQ::QuickUSBHandler, [68](#)
- kDataAddress
 - DAQ::QuickUSBHandler, [68](#)
- kFIFOConfig
 - DAQ::QuickUSBHandler, [68](#)
- kFPGAType
 - DAQ::QuickUSBHandler, [68](#)
- kGPIFA8gfiadr8
 - DAQ::QuickUSBHandler, [68](#)
- kGPIFA8gpio
 - DAQ::QuickUSBHandler, [68](#)
- kHandleACK
 - DAQ::QuickUSBHandler, [67](#)
- kHighLogic
 - DAQ::QuickUSBHandler, [68](#)
- kI2CBusClkSpeed100kHz
 - DAQ::QuickUSBHandler, [67](#)
- kI2CBusClkSpeed400kHz
 - DAQ::QuickUSBHandler, [67](#)
- kl2CTL
 - DAQ::QuickUSBHandler, [68](#)
- klgnoreACK
 - DAQ::QuickUSBHandler, [67](#)
- kLowLogic
 - DAQ::QuickUSBHandler, [68](#)
- kMISOPIN2
 - DAQ::QuickUSBHandler, [68](#)
- kMISOPIN5
 - DAQ::QuickUSBHandler, [68](#)
- kNCEPIN2
 - DAQ::QuickUSBHandler, [68](#)
- kNCEPIN7
 - DAQ::QuickUSBHandler, [68](#)
- kNoACK
 - DAQ::QuickUSBHandler, [67](#)
- kNormalCompletion
 - DAQ::QuickUSBHandler, [67](#)
- kPinFlags
 - DAQ::QuickUSBHandler, [68](#)
- kPortA
 - DAQ::QuickUSBHandler, [68](#)
- kPortAConfig
 - DAQ::QuickUSBHandler, [68](#)
- kPortB
 - DAQ::QuickUSBHandler, [68](#)
- kPortC
 - DAQ::QuickUSBHandler, [68](#)
- kPortD
 - DAQ::QuickUSBHandler, [68](#)
- kPortE
 - DAQ::QuickUSBHandler, [68](#)
- kSPICPHAclocksampler
 - DAQ::QuickUSBHandler, [68](#)
- kSPICPHAsampleclock
 - DAQ::QuickUSBHandler, [68](#)
- kSPICPOLinverted
 - DAQ::QuickUSBHandler, [68](#)
- kSPICPOLnormal
 - DAQ::QuickUSBHandler, [68](#)
- kSPIConfig
 - DAQ::QuickUSBHandler, [68](#)
- kSPIENDIANlsb
 - DAQ::QuickUSBHandler, [68](#)
- kSPIENDIANmsb
 - DAQ::QuickUSBHandler, [68](#)
- kSPIPORTA
 - DAQ::QuickUSBHandler, [68](#)
- kSPIPORTE
 - DAQ::QuickUSBHandler, [68](#)
- kSlaveFIFOFlags
 - DAQ::QuickUSBHandler, [68](#)
- kSlaveWait
 - DAQ::QuickUSBHandler, [67](#)
- kTimeout
 - DAQ::QuickUSBHandler, [67](#)
- kTimeoutHigh
 - DAQ::QuickUSBHandler, [68](#)

- kTimeoutLow
 - DAQ::QuickUSBHandler, 68
- kUSBFULLSpeedAllow
 - DAQ::QuickUSBHandler, 67
- kUSBFULLSpeedForce
 - DAQ::QuickUSBHandler, 67
- kVersionSpeed
 - DAQ::QuickUSBHandler, 68
- kWordWide
 - DAQ::QuickUSBHandler, 68
- Listen
 - Socket, 81
- LogRedirector, 42
 - contents, 42
 - fRedirect, 43
 - fSS, 43
 - LogRedirector, 42
- Logger, 41
 - ~Logger, 41
 - fBuffer, 41
 - fStream, 41
 - Logger, 41
- LogicLevel
 - DAQ::QuickUSBHandler, 67
- MASTER
 - Socket, 79
- magic
 - file_header_t, 28
- MajorVersion
 - DAQ::QuickUSBHandler::Version, 107
- Message, 43
 - ~Message, 44
 - Dump, 44
 - fString, 44
 - GetKey, 44
 - GetString, 44
 - IsFromWeb, 44
 - Message, 44
- Messenger, 45
 - ~Messenger, 47
 - AddClient, 47
 - Broadcast, 47
 - Connect, 48
 - Disconnect, 48
 - DisconnectClient, 48
 - fNumAttempts, 53
 - fPID, 53
 - fStderrPipe, 53
 - fStdoutPipe, 53
 - GetType, 50
 - Messenger, 46
 - ProcessMessage, 50
 - Receive, 50
 - Send, 51
 - SendAll, 51, 52
 - StartAcquisition, 52
 - StopAcquisition, 52
 - SwitchClientType, 52
 - MinorVersion
 - DAQ::QuickUSBHandler::Version, 107
 - NewBurst
 - OnlineDBHandler, 55
 - NewPlot
 - DQM::DQMProcess, 24
 - NewRun
 - OnlineDBHandler, 55
 - num_hptdc
 - file_header_t, 28
 - NumErrors
 - TDCMeasurement, 107
 - NumEvents
 - TDCMeasurement, 107
 - OLEADING
 - DAQ::TDC, 93
 - OTRAILING
 - DAQ::TDC, 93
 - Object
 - SocketMessage, 90
 - OnlineDBHandler, 53
 - ~OnlineDBHandler, 54
 - BuildTables, 55
 - BurstInfos, 54
 - fDB, 56
 - GetLastBurst, 55
 - GetLastRun, 55
 - GetRunInfo, 55
 - GetRuns, 55
 - GetTDCConditions, 55
 - NewBurst, 55
 - NewRun, 55
 - OnlineDBHandler, 54
 - RunCollection, 54
 - Select, 55
 - SetHVConditions, 55
 - SetTDCConditions, 56
 - TDCConditionsCollection, 54
 - OnlineDBHandler::BurstInfo, 15
 - burst_id, 15
 - time_start, 15
 - OnlineDBHandler::TDCConditions, 95
 - detector, 95
 - operator=, 95
 - operator==, 95
 - run_id, 95
 - tdc_acq_mode, 95
 - tdc_address, 95
 - tdc_det_mode, 95
 - tdc_id, 95
 - Open
 - FileReader, 31
 - OpenFile
 - DAQ::FPGAHandler, 35
 - operator<<
 - DAQ, 13

- DAQ::QuickUSBHandler, 77
- DAQ::QuickUSBHandler::FIFOFlags, 27
- TDCErrrorFlag, 97
- operator=
 - OnlineDBHandler::TDCConditions, 95
- operator==
 - OnlineDBHandler::TDCConditions, 95
- PAIR
 - DAQ::TDC, 93
- PPSCanvas
 - DQM::PPSCanvas, 58
- ParseMessage
 - Client, 19
 - DQM::DQMProcess, 25
- PrepareConnection
 - Socket, 81
- ProcessMessage
 - Messenger, 50
- QuarticCanvas
 - DQM::QuarticCanvas, 61, 62
- QuickUSBHandler
 - DAQ::QuickUSBHandler, 69
- RDY0
 - DAQ::QuickUSBHandler::FIFOFlags, 27
- RDY1
 - DAQ::QuickUSBHandler::FIFOFlags, 27
- ReadConfiguration
 - DAQ::TDC, 94
- ReadFIFOEmpty
 - DAQ::QuickUSBHandler::FIFOFlags, 27
- ReadFIFOFull
 - DAQ::QuickUSBHandler::FIFOFlags, 27
- ReadRegister
 - DAQ::TDC, 94
- ReadStatus
 - DAQ::TDC, 94
- Receive
 - Client, 19, 20
 - Messenger, 50
- RegisterTest
 - DAQ::FPGAHandler, 35
- Reset
 - DAQ::QuickUSBHandler, 73
- RetrieveSetupWord
 - DAQ::FPGAHandler, 35
- Run
 - DQM::DQMProcess, 25
- run_id
 - file_header_t, 28
 - OnlineDBHandler::TDCConditions, 95
- RunCollection
 - OnlineDBHandler, 54
- SPIConfig
 - DAQ::QuickUSBHandler, 68
- Save
 - DQM::GastofCanvas, 40
 - DQM::PPSCanvas, 59
 - DQM::QuarticCanvas, 63
- Select
 - OnlineDBHandler, 55
- SelectConnections
 - Socket, 82
- Send
 - Client, 20
 - Messenger, 51
- SendAll
 - Messenger, 51, 52
- SendAndReceive
 - Client, 21
- SendConfiguration
 - DAQ::TDC, 94
- SendMessage
 - Socket, 82
- SendSetupWord
 - DAQ::FPGAHandler, 35
- SetCPUConfig
 - DAQ::QuickUSBHandler, 73
- SetConfigRegister
 - DAQ::QuickUSBHandler, 73
- SetDataAddress
 - DAQ::QuickUSBHandler, 73
- SetEventsCollection
 - TDCMeasurement, 107
- SetFIFOConfig
 - DAQ::QuickUSBHandler, 74
- SetFPGAType
 - DAQ::QuickUSBHandler, 74
- SetHVConditions
 - OnlineDBHandler, 55
- SetI2CTL
 - DAQ::QuickUSBHandler, 74
- SetKeyValue
 - SocketMessage, 90, 91
- SetPort
 - DAQ::QuickUSBHandler, 75
 - Socket, 82
- SetRunInfo
 - DQM::GastofCanvas, 40
 - DQM::PPSCanvas, 59
 - DQM::QuarticCanvas, 63
- SetSPIConfig
 - DAQ::QuickUSBHandler, 75
- SetSetupRegister
 - DAQ::TDC, 94
- SetSocketId
 - Socket, 82
- SetTDCConditions
 - OnlineDBHandler, 56
- SetTDCSetup
 - DAQ::FPGAHandler, 35
- SetUpperLabel
 - DQM::GastofCanvas, 40
 - DQM::PPSCanvas, 59

- DQM::QuarticCanvas, 63
- SetWord
 - TDCEvent, 104
- SetWordWide
 - DAQ::QuickUSBHandler, 75
- SettingsRegister
 - DAQ::QuickUSBHandler, 68
- Socket, 77
 - ~Socket, 79
 - AcceptConnections, 79
 - Bind, 80
 - CLIENT, 79
 - Configure, 80
 - Create, 80
 - DAQ, 79
 - DETECTOR, 79
 - DQM, 79
 - DumpConnected, 80
 - fAddress, 83
 - fBuffer, 83
 - fMaster, 83
 - fPort, 83
 - fReadFds, 83
 - fSocketId, 83
 - fSocketsConnected, 83
 - FetchMessage, 80
 - GetPort, 81
 - GetSocketId, 81
 - GetSocketType, 81
 - INVALID, 79
 - IsWebSocket, 81
 - Listen, 81
 - MASTER, 79
 - PrepareConnection, 81
 - SelectConnections, 82
 - SendMessage, 82
 - SetPort, 82
 - SetSocketId, 82
 - Socket, 79
 - SocketCollection, 79
 - SocketType, 79
 - Start, 82
 - Stop, 83
 - WEBSOCKET_CLIENT, 79
- Socket communication objects, 9
- SocketCollection
 - Socket, 79
- SocketMessage, 84
 - ~SocketMessage, 89
 - Dump, 89
 - fMessage, 91
 - GetCleanedValue, 89
 - GetIntValue, 89
 - GetKey, 89
 - GetString, 89
 - GetValue, 89
 - GetVectorValue, 90
 - Object, 90
 - SetKeyValue, 90, 91
 - SocketMessage, 86–89
 - String, 91
- SocketType
 - Socket, 79
- SoftReset
 - DAQ::TDC, 94
- spill_id
 - file_header_t, 28
- Start
 - Socket, 82
- StartAcquisition
 - DAQ::FPGAHandler, 36
 - Messenger, 52
- StartBulkTransfer
 - DAQ::QuickUSBHandler, 76
- Stop
 - DAQ::FPGAHandler, 36
 - Socket, 83
- StopAcquisition
 - DAQ::FPGAHandler, 36
 - Messenger, 52
- StopBulkTransfer
 - DAQ::QuickUSBHandler, 76
- String
 - SocketMessage, 91
- SwitchClientType
 - Messenger, 52
- TDC
 - DAQ::TDC, 93
- TDCConditionsCollection
 - OnlineDBHandler, 54
- TDCError
 - TDCEvent, 99
- TDCErrorFlag, 95
 - ~TDCErrorFlag, 96
 - Dump, 96
 - fWord, 97
 - GetWord, 96
 - HasGroupError, 96
 - HasInternalChipError, 96
 - HasL1BufferOverflow, 97
 - HasReachedEventSizeLimit, 97
 - HasReadoutFIFOOverflow, 97
 - HasTriggerFIFOOverflow, 97
 - operator<<, 97
 - TDCErrorFlag, 96
- TDCEvent, 97
 - ~TDCEvent, 99
 - Dump, 99
 - ETTT, 99
 - EventType, 99
 - fWord, 104
 - Filler, 99
 - GetBunchId, 99
 - GetChannelId, 100
 - GetETTT, 100
 - GetErrorFlags, 100

- GetEventCount, [101](#)
- GetEventId, [101](#)
- GetGeo, [101](#)
- GetStatus, [102](#)
- GetTDCId, [102](#)
- GetTime, [102](#)
- GetType, [103](#)
- GetWidth, [103](#)
- GetWord, [103](#)
- GetWordCount, [103](#)
- GlobalHeader, [99](#)
- GlobalTrailer, [99](#)
- IsTrailing, [103](#)
- SetWord, [104](#)
- TDCError, [99](#)
- TDCEvent, [99](#)
- TDCHeader, [99](#)
- TDCMeasurement, [99](#)
- TDCTrailer, [99](#)
- Trigger, [99](#)
- TDCHeader
 - TDCEvent, [99](#)
- TDCMeasurement, [104](#)
 - ~TDCMeasurement, [105](#)
 - Dump, [105](#)
 - fEvents, [107](#)
 - fMap, [107](#)
 - GetBunchId, [106](#)
 - GetChannelId, [106](#)
 - GetETTT, [106](#)
 - GetEventId, [106](#)
 - GetLeadingTime, [106](#)
 - GetTDCId, [106](#)
 - GetToT, [106](#)
 - GetTrailingTime, [107](#)
 - NumErrors, [107](#)
 - NumEvents, [107](#)
 - SetEventsCollection, [107](#)
 - TDCEvent, [99](#)
 - TDCMeasurement, [105](#)
- TDCTrailer
 - TDCEvent, [99](#)
- TRAILHEAD
 - DAQ::TDC, [93](#)
- TRIG_MATCH
 - HPTDC chip control, [11](#)
- tdc_acq_mode
 - OnlineDBHandler::TDCConditions, [95](#)
- tdc_address
 - OnlineDBHandler::TDCConditions, [95](#)
- tdc_det_mode
 - OnlineDBHandler::TDCConditions, [95](#)
- tdc_id
 - OnlineDBHandler::TDCConditions, [95](#)
- time_start
 - OnlineDBHandler::BurstInfo, [15](#)
- Trigger
 - TDCEvent, [99](#)
- USBFullSpeed
 - DAQ::QuickUSBHandler, [69](#)
- USBHighSpeed
 - DAQ::QuickUSBHandler, [69](#)
- USBSpeed
 - DAQ::QuickUSBHandler, [68](#)
- UpdatedPlot
 - DQM::DQMProcess, [24](#)
- WEBSOCKET_CLIENT
 - Socket, [79](#)
- WordWide
 - DAQ::QuickUSBHandler, [69](#)
- Write
 - DAQ::QuickUSBHandler, [76](#)
- WriteFIFOEmpty
 - DAQ::QuickUSBHandler::FIFOFlags, [27](#)
- WriteFIFOFull
 - DAQ::QuickUSBHandler::FIFOFlags, [27](#)
- WriteRegister
 - DAQ::TDC, [94](#)
- x
 - DQM::GastofCanvas::Coord, [21](#)
 - DQM::QuarticCanvas::Coord, [22](#)
- XilinxSlaveSerial
 - DAQ::QuickUSBHandler, [67](#)
- y
 - DQM::GastofCanvas::Coord, [22](#)
 - DQM::QuarticCanvas::Coord, [22](#)