

2015 Test beam Run Control

Generated by Doxygen 1.8.9.1

Fri Apr 24 2015 02:04:49

Contents

1	Module Index	1
1.1	Modules	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Data Structure Index	5
3.1	Data Structures	5
4	Module Documentation	7
4.1	Socket communication objects	7
4.1.1	Detailed Description	7
4.1.2	Enumeration Type Documentation	7
4.1.2.1	SocketType	7
4.2	HPTDC chip control	9
4.2.1	Detailed Description	10
4.2.2	Enumeration Type Documentation	10
4.2.2.1	CoreClockSource	10
4.2.2.2	DeadTime	10
4.2.2.3	DLLClockSource	10
4.2.2.4	DLLSpeedMode	10
4.2.2.5	EdgeResolution	10
4.2.2.6	EnabledError	11
4.2.2.7	EventType	11
4.2.2.8	IOClockSource	11
4.2.2.9	ReadoutSingleCycleSpeed	11
4.2.2.10	ReadoutSpeed	12
4.2.2.11	SerialClockSource	12
4.2.2.12	SerialStrobeType	12
4.2.2.13	WidthResolution	12
5	Data Structure Documentation	13
5.1	Client Class Reference	13

5.1.1	Detailed Description	14
5.1.2	Constructor & Destructor Documentation	15
5.1.2.1	Client	15
5.1.2.2	Client	15
5.1.2.3	~Client	15
5.1.3	Member Function Documentation	15
5.1.3.1	Announce	15
5.1.3.2	Connect	16
5.1.3.3	Disconnect	16
5.1.3.4	GetType	17
5.1.3.5	ParseMessage	17
5.1.3.6	Receive	17
5.1.3.7	Send	18
5.1.4	Field Documentation	18
5.1.4.1	fClientId	18
5.1.4.2	flsConnected	18
5.2	Exception Class Reference	18
5.2.1	Detailed Description	19
5.2.2	Constructor & Destructor Documentation	19
5.2.2.1	Exception	19
5.2.2.2	Exception	19
5.2.2.3	~Exception	19
5.2.3	Member Function Documentation	19
5.2.3.1	Description	19
5.2.3.2	Dump	20
5.2.3.3	ErrorNumber	20
5.2.3.4	From	20
5.2.3.5	Type	20
5.2.3.6	TypeString	20
5.2.4	Field Documentation	20
5.2.4.1	fDescription	20
5.2.4.2	fErrorNumber	20
5.2.4.3	fFrom	20
5.2.4.4	fType	20
5.3	file_header_t Struct Reference	21
5.3.1	Detailed Description	21
5.3.2	Field Documentation	21
5.3.2.1	config	21
5.3.2.2	magic	21
5.3.2.3	run_id	22

5.3.2.4	spill_id	22
5.4	FPGAHandler Class Reference	22
5.4.1	Detailed Description	23
5.4.2	Constructor & Destructor Documentation	24
5.4.2.1	FPGAHandler	24
5.4.2.2	~FPGAHandler	24
5.4.3	Member Function Documentation	24
5.4.3.1	CloseFile	24
5.4.3.2	GetConfiguration	24
5.4.3.3	GetFilename	25
5.4.3.4	GetType	25
5.4.3.5	OpenFile	25
5.4.3.6	ReadBoundaryScanRegister	25
5.4.3.7	ReadBuffer	25
5.4.3.8	ReadConfiguration	25
5.4.3.9	SendConfiguration	25
5.4.3.10	SetConfiguration	26
5.4.4	Field Documentation	26
5.4.4.1	fFilename	26
5.4.4.2	fIsFileOpen	26
5.4.4.3	fIsTDCInReadout	26
5.4.4.4	fOutput	26
5.4.4.5	fTDCBSR	26
5.4.4.6	fTDCSetup	26
5.5	HTTPMessage Class Reference	27
5.5.1	Detailed Description	28
5.5.2	Constructor & Destructor Documentation	28
5.5.2.1	HTTPMessage	28
5.5.2.2	HTTPMessage	28
5.5.3	Member Function Documentation	29
5.5.3.1	Decode	29
5.5.3.2	Dump	29
5.5.3.3	Encode	29
5.5.3.4	GetKey	29
5.5.4	Field Documentation	29
5.5.4.1	fOriginalString	29
5.5.4.2	fWS	29
5.6	ListenerInfo Struct Reference	29
5.6.1	Detailed Description	29
5.6.2	Field Documentation	29

5.6.2.1	name	29
5.6.2.2	type	29
5.7	Message Class Reference	29
5.7.1	Detailed Description	30
5.7.2	Constructor & Destructor Documentation	31
5.7.2.1	Message	31
5.7.2.2	Message	31
5.7.2.3	Message	31
5.7.2.4	~Message	31
5.7.3	Member Function Documentation	31
5.7.3.1	Dump	31
5.7.3.2	GetKey	31
5.7.3.3	GetString	31
5.7.3.4	IsFromWeb	31
5.7.4	Field Documentation	31
5.7.4.1	fString	31
5.8	Messenger Class Reference	31
5.8.1	Detailed Description	33
5.8.2	Constructor & Destructor Documentation	33
5.8.2.1	Messenger	33
5.8.2.2	Messenger	33
5.8.2.3	~Messenger	34
5.8.3	Member Function Documentation	34
5.8.3.1	AddClient	34
5.8.3.2	Broadcast	34
5.8.3.3	Connect	35
5.8.3.4	Disconnect	35
5.8.3.5	DisconnectClient	36
5.8.3.6	GetType	36
5.8.3.7	ProcessMessage	36
5.8.3.8	Receive	37
5.8.3.9	Send	37
5.8.3.10	SwitchClientType	38
5.8.4	Field Documentation	38
5.8.4.1	fListenersInfo	38
5.8.4.2	fNumAttempts	38
5.8.4.3	fWS	38
5.9	Socket Class Reference	39
5.9.1	Detailed Description	40
5.9.2	Member Typedef Documentation	41

5.9.2.1	SocketCollection	41
5.9.3	Constructor & Destructor Documentation	41
5.9.3.1	Socket	41
5.9.3.2	Socket	41
5.9.3.3	~Socket	41
5.9.4	Member Function Documentation	41
5.9.4.1	AcceptConnections	41
5.9.4.2	Bind	41
5.9.4.3	Configure	42
5.9.4.4	Create	42
5.9.4.5	DumpConnected	42
5.9.4.6	FetchMessage	42
5.9.4.7	GetPort	42
5.9.4.8	GetSocketId	42
5.9.4.9	GetSocketType	42
5.9.4.10	IsWebSocket	42
5.9.4.11	Listen	42
5.9.4.12	PrepareConnection	43
5.9.4.13	SelectConnections	43
5.9.4.14	SendMessage	43
5.9.4.15	SetPort	43
5.9.4.16	SetSocketId	43
5.9.4.17	Start	44
5.9.4.18	Stop	44
5.9.5	Field Documentation	44
5.9.5.1	fAddress	44
5.9.5.2	fBuffer	44
5.9.5.3	fMaster	44
5.9.5.4	fPort	44
5.9.5.5	fReadFds	44
5.9.5.6	fSocketId	44
5.9.5.7	fSocketsConnected	44
5.10	SocketMessage Class Reference	45
5.10.1	Detailed Description	46
5.10.2	Constructor & Destructor Documentation	47
5.10.2.1	SocketMessage	47
5.10.2.2	SocketMessage	47
5.10.2.3	SocketMessage	47
5.10.2.4	SocketMessage	47
5.10.2.5	SocketMessage	47

5.10.2.6	SocketMessage	48
5.10.2.7	SocketMessage	48
5.10.2.8	SocketMessage	48
5.10.2.9	SocketMessage	48
5.10.2.10	SocketMessage	49
5.10.2.11	SocketMessage	49
5.10.2.12	~SocketMessage	49
5.10.3	Member Function Documentation	49
5.10.3.1	Dump	49
5.10.3.2	GetIntValue	49
5.10.3.3	GetKey	50
5.10.3.4	GetString	50
5.10.3.5	GetValue	50
5.10.3.6	GetVectorValue	50
5.10.3.7	Object	50
5.10.3.8	SetKeyValue	50
5.10.3.9	SetKeyValue	50
5.10.3.10	SetKeyValue	51
5.10.3.11	SetKeyValue	51
5.10.3.12	String	51
5.10.4	Field Documentation	51
5.10.4.1	fMessage	51
5.11	TDCBoundaryScanRegister Class Reference	51
5.11.1	Detailed Description	52
5.11.2	Constructor & Destructor Documentation	52
5.11.2.1	TDCBoundaryScanRegister	52
5.11.2.2	~TDCBoundaryScanRegister	53
5.12	TDCCControl Class Reference	53
5.12.1	Member Enumeration Documentation	54
5.12.1.1	EnablePattern	54
5.12.2	Constructor & Destructor Documentation	54
5.12.2.1	TDCCControl	54
5.12.2.2	TDCCControl	54
5.12.2.3	~TDCCControl	54
5.12.3	Member Function Documentation	54
5.12.3.1	Dump	54
5.12.3.2	SetControlParity	55
5.12.3.3	SetDLLReset	55
5.12.3.4	SetEnablePattern	55
5.12.3.5	SetGlobalReset	55

5.12.3.6	SetPLLReset	56
5.12.4	Field Documentation	56
5.12.4.1	kControlParity	56
5.12.4.2	kDLLReset	56
5.12.4.3	kEnableChannel	56
5.12.4.4	kEnablePattern	56
5.12.4.5	kGlobalReset	56
5.12.4.6	kPLLReset	56
5.13	TDCEvent Class Reference	56
5.13.1	Detailed Description	57
5.13.2	Constructor & Destructor Documentation	57
5.13.2.1	TDCEvent	57
5.13.2.2	~TDCEvent	57
5.13.3	Member Function Documentation	57
5.13.3.1	GetBunchId	57
5.13.3.2	GetErrorFlags	58
5.13.3.3	GetEventId	58
5.13.3.4	GetLeadingTime	58
5.13.3.5	GetTDCId	58
5.13.3.6	GetTrailingTime	59
5.13.3.7	GetType	59
5.13.3.8	GetWidth	59
5.13.3.9	GetWordCount	59
5.13.4	Field Documentation	59
5.13.4.1	fWord	60
5.14	TDCRegister Class Reference	60
5.14.1	Detailed Description	61
5.14.2	Member Typedef Documentation	61
5.14.2.1	bit	61
5.14.2.2	word_t	61
5.14.3	Constructor & Destructor Documentation	61
5.14.3.1	TDCRegister	61
5.14.3.2	~TDCRegister	61
5.14.4	Member Function Documentation	61
5.14.4.1	DumpRegister	61
5.14.4.2	GetBits	61
5.14.4.3	GetNumWords	61
5.14.4.4	GetWord	62
5.14.4.5	SetBits	62
5.14.4.6	SetWord	62

5.14.5	Field Documentation	62
5.14.5.1	fNumWords	62
5.14.5.2	fWord	62
5.15	TDCSetup Class Reference	62
5.15.1	Detailed Description	69
5.15.2	Constructor & Destructor Documentation	70
5.15.2.1	TDCSetup	70
5.15.2.2	TDCSetup	71
5.15.2.3	~TDCSetup	71
5.15.3	Member Function Documentation	71
5.15.3.1	Dump	72
5.15.3.2	GetChannelOffset	72
5.15.3.3	GetCoarseCountOffset	72
5.15.3.4	GetDeadTime	73
5.15.3.5	GetDLLAdjustment	73
5.15.3.6	GetEdgeResolution	74
5.15.3.7	GetEdgesPairing	74
5.15.3.8	GetEnableError	74
5.15.3.9	GetEnableErrorBypass	75
5.15.3.10	GetEnableErrorMark	75
5.15.3.11	GetEnableJTAGReadout	75
5.15.3.12	GetEnableReadoutOccupancy	76
5.15.3.13	GetEnableReadoutSeparator	76
5.15.3.14	GetEnableSerial	76
5.15.3.15	GetLeadingMode	76
5.15.3.16	GetMatchWindow	77
5.15.3.17	GetMaxEventSize	77
5.15.3.18	GetRCAdjustment	77
5.15.3.19	GetReadoutFIFOSize	78
5.15.3.20	GetRejectCountOffset	78
5.15.3.21	GetRejectFIFOFull	78
5.15.3.22	GetSearchWindow	79
5.15.3.23	GetSetupParity	79
5.15.3.24	GetTestInvert	80
5.15.3.25	GetTestMode	80
5.15.3.26	GetTrailingMode	80
5.15.3.27	GetTriggerCountOffset	80
5.15.3.28	GetTriggerLatency	81
5.15.3.29	GetTriggerMatchingMode	81
5.15.3.30	GetVernierOffset	81

5.15.3.31 GetWidthResolution	82
5.15.3.32 SetAllChannelsOffset	82
5.15.3.33 SetAllTapsDLLAdjustment	82
5.15.3.34 SetBypassInputs	83
5.15.3.35 SetChannelOffset	83
5.15.3.36 SetCoarseCountOffset	83
5.15.3.37 SetConstantValues	83
5.15.3.38 SetCoreClockDelay	84
5.15.3.39 SetCoreClockSource	85
5.15.3.40 SetDeadTime	85
5.15.3.41 SetDLLAdjustment	85
5.15.3.42 SetDLLClockDelay	86
5.15.3.43 SetDLLClockSource	86
5.15.3.44 SetDLLControl	86
5.15.3.45 SetDLLMode	87
5.15.3.46 SetEdgeResolution	87
5.15.3.47 SetEdgesPairing	87
5.15.3.48 SetEnableAutomaticReject	88
5.15.3.49 SetEnableBytewise	88
5.15.3.50 SetEnableDirectBunchReset	88
5.15.3.51 SetEnableDirectEventReset	89
5.15.3.52 SetEnableDirectTrigger	89
5.15.3.53 SetEnableError	89
5.15.3.54 SetEnableErrorBypass	90
5.15.3.55 SetEnableErrorMark	90
5.15.3.56 SetEnableGlobalHeader	90
5.15.3.57 SetEnableGlobalTrailer	91
5.15.3.58 SetEnableJTAGReadout	91
5.15.3.59 SetEnableLocalHeader	91
5.15.3.60 SetEnableLocalTrailer	92
5.15.3.61 SetEnableMasterResetCode	92
5.15.3.62 SetEnableMasterResetOnEventReset	92
5.15.3.63 SetEnableOverflowDetect	93
5.15.3.64 SetEnableReadoutOccupancy	93
5.15.3.65 SetEnableReadoutSeparator	93
5.15.3.66 SetEnableRelative	94
5.15.3.67 SetEnableResetChannelBufferWhenSeparator	94
5.15.3.68 SetEnableSeparatorOnBunchReset	94
5.15.3.69 SetEnableSeparatorOnEventReset	95
5.15.3.70 SetEnableSerial	95

5.15.3.71 SetEnableSetCountersOnBunchReset	95
5.15.3.72 SetEnableTTLClock	96
5.15.3.73 SetEnableTTLControl	96
5.15.3.74 SetEnableTTLHit	97
5.15.3.75 SetEnableTTLReset	97
5.15.3.76 SetEnableTTLSerial	97
5.15.3.77 SetEventCountOffset	98
5.15.3.78 SetIOClockDelay	98
5.15.3.79 SetIOClockSource	98
5.15.3.80 SetKeepToken	99
5.15.3.81 SetLeadingMode	99
5.15.3.82 SetLowPowerMode	99
5.15.3.83 SetMaster	100
5.15.3.84 SetMatchWindow	100
5.15.3.85 SetMaxEventSize	100
5.15.3.86 SetModeRC	101
5.15.3.87 SetModeRCCompression	101
5.15.3.88 SetPLLControl	101
5.15.3.89 SetRCAdjustment	102
5.15.3.90 SetReadoutFIFOSize	102
5.15.3.91 SetReadoutSingleCycleSpeed	102
5.15.3.92 SetReadoutSpeedSelect	103
5.15.3.93 SetRejectCountOffset	103
5.15.3.94 SetRejectFIFOFull	104
5.15.3.95 SetRollOver	104
5.15.3.96 SetSearchWindow	104
5.15.3.97 SetSerialClockDelay	104
5.15.3.98 SetSerialClockSource	105
5.15.3.99 SetSerialDelay	105
5.15.3.100SetSetupParity	105
5.15.3.101SetStrobeSelect	106
5.15.3.102SetTestInvert	106
5.15.3.103SetTestMode	106
5.15.3.104SetTokenDelay	107
5.15.3.105SetTrailingMode	107
5.15.3.106SetTriggerCountOffset	107
5.15.3.107SetTriggerMatchingMode	108
5.15.3.108SetVernierOffset	108
5.15.3.109SetWidthResolution	108
5.15.4 Field Documentation	109

5.15.4.1	kCoarseCountOffset	109
5.15.4.2	kCoreClockDelay	109
5.15.4.3	kCoreClockSource	109
5.15.4.4	kDeadTime	109
5.15.4.5	kDLLClockDelay	109
5.15.4.6	kDLLClockSource	109
5.15.4.7	kDLLControl	109
5.15.4.8	kDLLMode	109
5.15.4.9	kDLLTapAdjust0	109
5.15.4.10	kEnableAutomaticReject	109
5.15.4.11	kEnableBytewise	109
5.15.4.12	kEnableDirectBunchReset	109
5.15.4.13	kEnableDirectEventReset	109
5.15.4.14	kEnableDirectTrigger	109
5.15.4.15	kEnableError	109
5.15.4.16	kEnableErrorBypass	109
5.15.4.17	kEnableErrorMark	109
5.15.4.18	kEnableGlobalHeader	109
5.15.4.19	kEnableGlobalTrailer	109
5.15.4.20	kEnableJTAGReadout	109
5.15.4.21	kEnableLocalHeader	109
5.15.4.22	kEnableLocalTrailer	110
5.15.4.23	kEnableMasterResetCode	110
5.15.4.24	kEnableMasterResetOnEventReset	110
5.15.4.25	kEnableMatching	110
5.15.4.26	kEnableOverflowDetect	110
5.15.4.27	kEnablePair	110
5.15.4.28	kEnableReadoutOccupancy	110
5.15.4.29	kEnableReadoutSeparator	110
5.15.4.30	kEnableRelative	110
5.15.4.31	kEnableResetChannelBufferWhenSeparator	110
5.15.4.32	kEnableSeparatorOnBunchReset	110
5.15.4.33	kEnableSeparatorOnEventReset	110
5.15.4.34	kEnableSerial	110
5.15.4.35	kEnableSetCountersOnBunchReset	110
5.15.4.36	kEnableTTLClock	110
5.15.4.37	kEnableTTLControl	110
5.15.4.38	kEnableTTLHit	110
5.15.4.39	kEnableTTLReset	110
5.15.4.40	kEnableTTLSerial	110

5.15.4.41 kEventCountOffset	110
5.15.4.42 kIOClockDelay	110
5.15.4.43 kIOClockSource	110
5.15.4.44 kKeepToken	110
5.15.4.45 kLeading	110
5.15.4.46 kLeadingResolution	110
5.15.4.47 kLowPowerMode	110
5.15.4.48 kMaster	110
5.15.4.49 kMatchWindow	110
5.15.4.50 kMaxEventSize	111
5.15.4.51 kModeRC	111
5.15.4.52 kModeRCCompression	111
5.15.4.53 kOffset0	111
5.15.4.54 kPLLControl	111
5.15.4.55 kRCAdjust0	111
5.15.4.56 kReadoutFIFOSize	111
5.15.4.57 kReadoutSingleCycleSpeed	111
5.15.4.58 kReadoutSpeedSelect	111
5.15.4.59 kRejectCountOffset	111
5.15.4.60 kRejectFIFOFull	111
5.15.4.61 kRollOver	111
5.15.4.62 kSearchWindow	111
5.15.4.63 kSelectBypassInputs	111
5.15.4.64 kSerialClockDelay	111
5.15.4.65 kSerialClockSource	111
5.15.4.66 kSerialDelay	111
5.15.4.67 kSetupParity	111
5.15.4.68 kStrobeSelect	111
5.15.4.69 kTDCId	111
5.15.4.70 kTestInvert	111
5.15.4.71 kTestMode	111
5.15.4.72 kTestSelect	111
5.15.4.73 kTokenDelay	111
5.15.4.74 kTrailing	111
5.15.4.75 kTriggerCountOffset	111
5.15.4.76 kVernierOffset	111
5.15.4.77 kWidthSelect	112
5.16 USBHandler Class Reference	112
5.16.1 Detailed Description	112
5.16.2 Constructor & Destructor Documentation	113

5.16.2.1	USBHandler	113
5.16.2.2	~USBHandler	113
5.16.3	Member Function Documentation	113
5.16.3.1	DumpDevice	113
5.16.3.2	Fetch	113
5.16.3.3	Init	113
5.16.3.4	Write	113
5.16.4	Field Documentation	113
5.16.4.1	fDevice	113
5.16.4.2	fHandle	113
Index		115

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Socket communication objects	7
HPTDC chip control	9

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Exception	18
file_header_t	21
ListenerInfo	29
Message	29
HTTPMessage	27
SocketMessage	45
Socket	39
Client	13
FPGAHandler	22
Messenger	31
TDCEvent	56
TDCRegister	60
TDCBoundaryScanRegister	51
TDCControl	53
TDCSetup	62
USBHandler	112
FPGAHandler	22

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Client	Base client object for the socket	13
Exception	A simple exception handler	18
file_header_t	Header to the output files	21
FPGAHandler	Driver for timing detectors' FPGA readout	22
HTTPMessage	Message to be transmitted through a WebSocket protocol	27
ListenerInfo	Information on a socket's listener	29
Message	Base socket message type	29
Messenger	Base master object for the socket	31
Socket	Base socket object from which clients/master from a socket inherit	39
SocketMessage	Socket-passed message type	45
TDCBoundaryScanRegister	51
TDCControl	53
TDCEvent	HPTDC event parser	56
TDCRegister	General register object to interact with a HPTDC chip	60
TDCSetup	Setup word to be sent to the HPTDC chip	62
USBHandler	Generic USB communication handler	112

Chapter 4

Module Documentation

4.1 Socket communication objects

Data Structures

- class [Client](#)
Base client object for the socket.
- class [HTTPMessage](#)
Message to be transmitted through a WebSocket protocol.
- struct [ListenerInfo](#)
Information on a socket's listener.
- class [Messenger](#)
Base master object for the socket.
- class [Socket](#)
Base socket object from which clients/master from a socket inherit.
- class [SocketMessage](#)
Socket-passed message type.

Enumerations

- enum [Socket::SocketType](#) {
[Socket::INVALID](#) = -1, [Socket::MASTER](#) = 0, [Socket::WEBSOCKET_CLIENT](#), [Socket::CLIENT](#),
[Socket::DETECTOR](#) }
Type of actor playing a role on the socket.

4.1.1 Detailed Description

4.1.2 Enumeration Type Documentation

4.1.2.1 enum [Socket::SocketType](#)

Type of actor playing a role on the socket.

Enumerator

INVALID

MASTER

WEBSOCKET_CLIENT

CLIENT
DETECTOR

4.2 HPTDC chip control

Data Structures

- class [TDCBoundaryScanRegister](#)
- class [TDCEvent](#)
HPTDC event parser.
- class [TDCRegister](#)
General register object to interact with a HPTDC chip.
- class [TDCSetup](#)
Setup word to be sent to the HPTDC chip.

Enumerations

- enum [TDCEvent::EventType](#) {
[TDCEvent::Invalid](#) = -1, [TDCEvent::GroupHeader](#) = 0, [TDCEvent::GroupTrailer](#), [TDCEvent::TDCHeader](#),
[TDCEvent::TDCTrailer](#), [TDCEvent::LeadingEdge](#), [TDCEvent::TrailingEdge](#), [TDCEvent::Error](#),
[TDCEvent::Debug](#) }
- enum [TDCSetup::EdgeResolution](#) {
[TDCSetup::E_100ps](#) = 0, [TDCSetup::E_200ps](#), [TDCSetup::E_400ps](#), [TDCSetup::E_800ps](#),
[TDCSetup::E_1p6ns](#), [TDCSetup::E_3p12ns](#), [TDCSetup::E_6p25ns](#), [TDCSetup::E_12p5ns](#) }
- enum [TDCSetup::DeadTime](#) { [TDCSetup::DT_5ns](#) = 0, [TDCSetup::DT_10ns](#), [TDCSetup::DT_30ns](#), [TDCSetup::DT_100ns](#) }
- enum [TDCSetup::WidthResolution](#) {
[TDCSetup::W_100ps](#) = 0, [TDCSetup::W_200ps](#), [TDCSetup::W_400ps](#), [TDCSetup::W_800ps](#),
[TDCSetup::W_1p6ns](#), [TDCSetup::W_3p2ns](#), [TDCSetup::W_6p25ns](#), [TDCSetup::W_12p5ns](#),
[TDCSetup::W_25ns](#), [TDCSetup::W_50ns](#), [TDCSetup::W_100ns](#), [TDCSetup::W_200ns](#),
[TDCSetup::W_400ns](#), [TDCSetup::W_800ns](#) }
- enum [TDCSetup::EnabledError](#) {
[TDCSetup::VernierError](#) = 0x1, [TDCSetup::CoarseError](#) = 0x2, [TDCSetup::ChannelSelectError](#) = 0x4, [TDCSetup::L1BufferParityError](#) = 0x8,
[TDCSetup::TriggerFIFOParityError](#) = 0x10, [TDCSetup::TriggerMatchingError](#) = 0x20, [TDCSetup::ReadoutFIFOParityError](#) = 0x40, [TDCSetup::ReadoutStateError](#) = 0x80,
[TDCSetup::SetupParityError](#) = 0x100, [TDCSetup::ControlParityError](#) = 0x200, [TDCSetup::JTAGInstructionParityError](#) = 0x400 }
- enum [TDCSetup::DLLSpeedMode](#) { [TDCSetup::DLL_40MHz](#) = 0x0, [TDCSetup::DLL_160MHz](#) = 0x1, [TDCSetup::DLL_320MHz](#) = 0x2, [TDCSetup::DLL_Illegal](#) = 0x3 }
- enum [TDCSetup::SerialClockSource](#) { [TDCSetup::Serial_pll_clock_80](#) = 0x0, [TDCSetup::Serial_pll_clock_160](#) = 0x1, [TDCSetup::Serial_pll_clock_40](#) = 0x2, [TDCSetup::Serial_aux_clock](#) = 0x3 }
- enum [TDCSetup::IOClockSource](#) { [TDCSetup::IO_clock_40](#) = 0x0, [TDCSetup::IO_pll_clock_80](#) = 0x1, [TDCSetup::IO_pll_clock_160](#) = 0x2, [TDCSetup::IO_aux_clock](#) = 0x3 }
- enum [TDCSetup::CoreClockSource](#) { [TDCSetup::Core_clock_40](#) = 0x0, [TDCSetup::Core_pll_clock_80](#) = 0x1, [TDCSetup::Core_pll_clock_160](#) = 0x2, [TDCSetup::Core_aux_clock](#) = 0x3 }
- enum [TDCSetup::DLLClockSource](#) {
[TDCSetup::DLL_clock_40](#) = 0x0, [TDCSetup::DLL_pll_clock_40](#) = 0x1, [TDCSetup::DLL_pll_clock_160](#) = 0x2,
[TDCSetup::DLL_pll_clock_320](#) = 0x3,
[TDCSetup::DLL_aux_clock](#) = 0x4 }
- enum [TDCSetup::ReadoutSpeed](#) { [TDCSetup::RO_Fixed](#) = 0x0, [TDCSetup::RO_pll_80Mbits_s](#) = 0x1 }
- enum [TDCSetup::SerialStrobeType](#) { [TDCSetup::SS_NoStrobe](#) = 0x0, [TDCSetup::SS_DSStrobe](#) = 0x1, [TDCSetup::SS_LeadingTrailingStrobe](#) = 0x2, [TDCSetup::SS_LeadingEdge](#) = 0x3 }
- enum [TDCSetup::ReadoutSingleCycleSpeed](#) {
[TDCSetup::RSC_40Mbits_s](#) = 0x0, [TDCSetup::RSC_20Mbits_s](#) = 0x1, [TDCSetup::RSC_10Mbits_s](#) = 0x2, [TDCSetup::RSC_5Mbits_s](#) = 0x3,
[TDCSetup::RSC_2p5Mbits_s](#) = 0x4, [TDCSetup::RSC_1p25Mbits_s](#) = 0x5, [TDCSetup::RSC_625kbits_s](#) = 0x6,
[TDCSetup::RSC_312p5kbits_s](#) = 0x7 }

4.2.1 Detailed Description

4.2.2 Enumeration Type Documentation

4.2.2.1 enum TDCSetup::CoreClockSource

Enumerator

Core_clock_40
Core_pll_clock_80
Core_pll_clock_160
Core_aux_clock

4.2.2.2 enum TDCSetup::DeadTime

Enumerator

DT_5ns
DT_10ns
DT_30ns
DT_100ns

4.2.2.3 enum TDCSetup::DLLClockSource

Enumerator

DLL_clock_40
DLL_pll_clock_40
DLL_pll_clock_160
DLL_pll_clock_320
DLL_aux_clock

4.2.2.4 enum TDCSetup::DLLSpeedMode

Enumerator

DLL_40MHz
DLL_160MHz
DLL_320MHz
DLL_Illegal

4.2.2.5 enum TDCSetup::EdgeResolution

Enumerator

E_100ps
E_200ps
E_400ps
E_800ps
E_1p6ns
E_3p12ns
E_6p25ns
E_12p5ns

4.2.2.6 enum TDCSetup::EnabledError

Enumerator

VernierError
CoarseError
ChannelSelectError
L1BufferParityError
TriggerFIFOParityError
TriggerMatchingError
ReadoutFIFOParityError
ReadoutStateError
SetupParityError
ControlParityError
JTAGInstructionParityError

4.2.2.7 enum TDCEvent::EventType

Enumerator

Invalid
GroupHeader
GroupTrailer
TDCHeader
TDCTrailer
LeadingEdge
TrailingEdge
Error
Debug

4.2.2.8 enum TDCSetup::IOClockSource

Enumerator

IO_clock_40
IO_pll_clock_80
IO_pll_clock_160
IO_aux_clock

4.2.2.9 enum TDCSetup::ReadoutSingleCycleSpeed

Enumerator

RSC_40Mbits_s
RSC_20Mbits_s
RSC_10Mbits_s
RSC_5Mbits_s
RSC_2p5Mbits_s
RSC_1p25Mbits_s
RSC_625kbits_s
RSC_312p5kbits_s

4.2.2.10 enum TDCSetup::ReadoutSpeed

Enumerator

RO_Fixed
RO_pll_80Mbits_s

4.2.2.11 enum TDCSetup::SerialClockSource

Enumerator

Serial_pll_clock_80
Serial_pll_clock_160
Serial_pll_clock_40
Serial_aux_clock

4.2.2.12 enum TDCSetup::SerialStrobeType

Enumerator

SS_NoStrobe
SS_DSStrobe
SS_LeadingTrailingStrobe
SS_LeadingEdge

4.2.2.13 enum TDCSetup::WidthResolution

Enumerator

W_100ps
W_200ps
W_400ps
W_800ps
W_1p6ns
W_3p2ns
W_6p25ns
W_12p5ns
W_25ns
W_50ns
W_100ns
W_200ns
W_400ns
W_800ns

Chapter 5

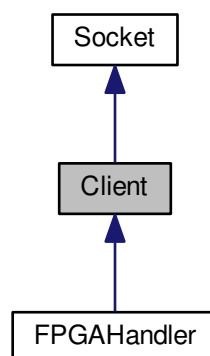
Data Structure Documentation

5.1 Client Class Reference

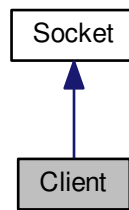
Base client object for the socket.

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

Inheritance diagram for Client:



Collaboration diagram for Client:



Public Member Functions

- [Client](#) ()
General void client constructor.
- [Client](#) (int port)
Bind a socket client to a given port.
- virtual [~Client](#) ()
- bool [Connect](#) ()
Bind this client to the socket.
- void [Disconnect](#) ()
Unbind this client from the socket.
- void [Send](#) (const [Message](#) &m) const
Send a message to the master through the socket.
- void [Receive](#) ()
Receive a socket message from the master.
- virtual void [ParseMessage](#) (const [SocketMessage](#) &m)
Parse a [SocketMessage](#) received from the master.
- virtual [SocketType](#) [GetType](#) () const
[Socket](#) actor type retrieval method.

Private Member Functions

- void [Announce](#) ()
Announce our entry on the socket to its master.

Private Attributes

- int [fClientId](#)
- bool [fIsConnected](#)

Additional Inherited Members

5.1.1 Detailed Description

Base client object for the socket.

[Client](#) object used by the server to send/receive commands from the messenger/broadcaster.

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

24 Mar 2015

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Client::Client () [inline]

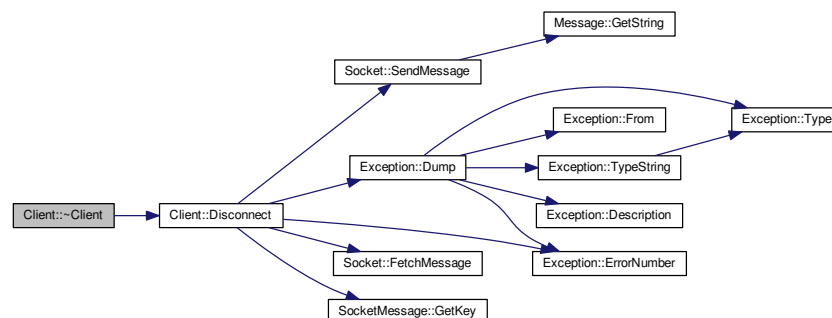
General void client constructor.

5.1.2.2 Client::Client (int *port*)

Bind a socket client to a given port.

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

Here is the call graph for this function:

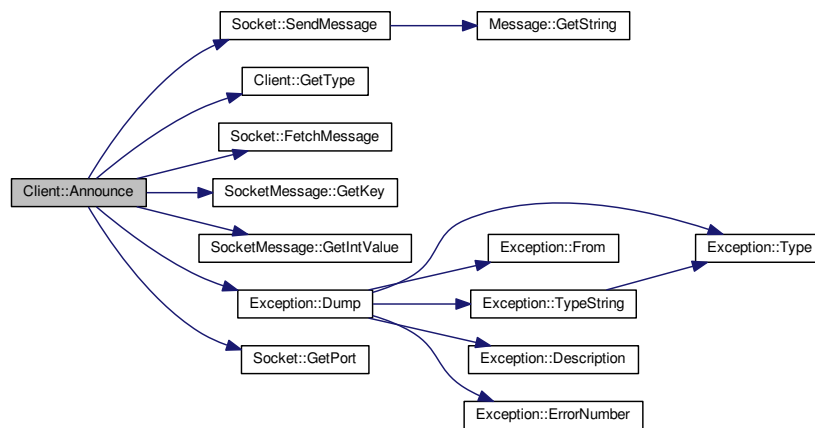


5.1.3 Member Function Documentation

5.1.3.1 void Client::Announce () [private]

Announce our entry on the socket to its master.

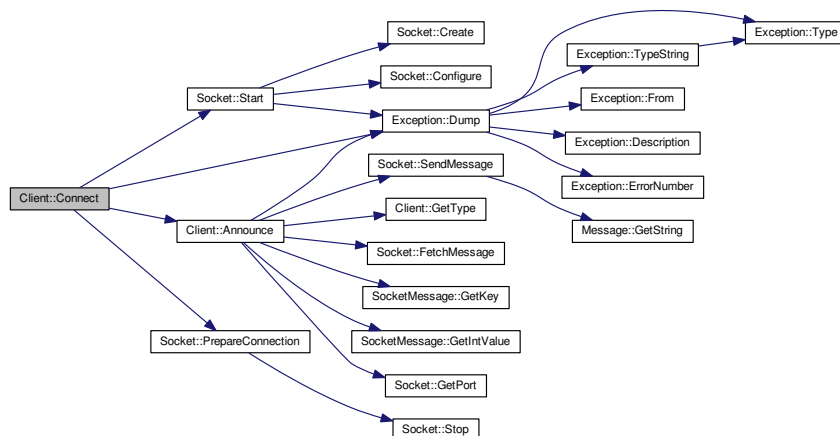
Here is the call graph for this function:



5.1.3.2 bool Client::Connect ()

Bind this client to the socket.

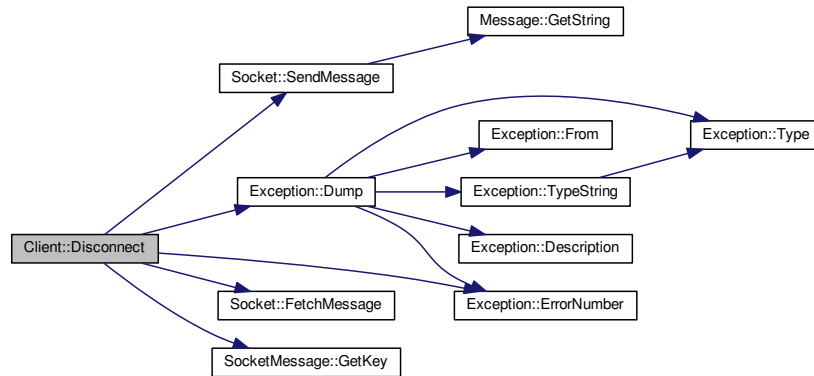
Here is the call graph for this function:



5.1.3.3 void Client::Disconnect ()

Unbind this client from the socket.

Here is the call graph for this function:



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

Socket actor type retrieval method.

Reimplemented in [FPGAHandler](#).

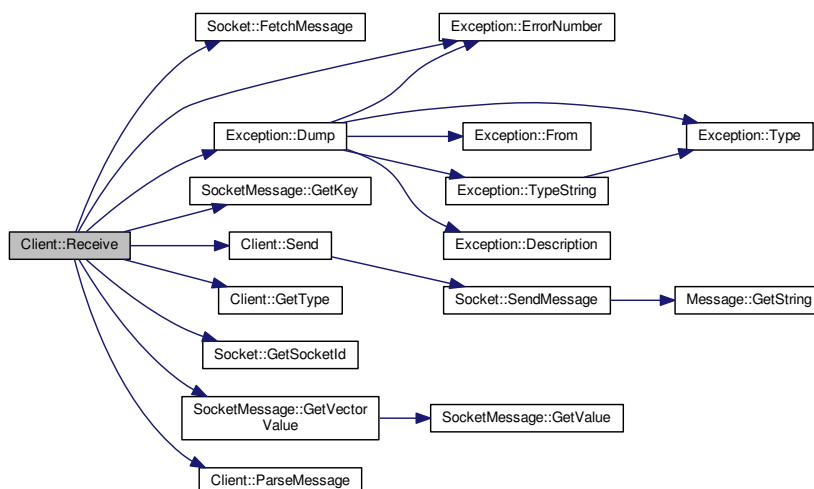
5.1.3.5 virtual void Client::ParseMessage (const SocketMessage & m) [inline],[virtual]

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

5.1.3.6 void Client::Receive ()

Receive a socket message from the master.

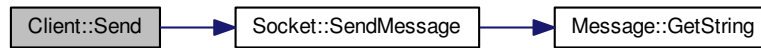
Here is the call graph for this function:



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

Send a message to the master through the socket.

Here is the call graph for this function:



5.1.4 Field Documentation

5.1.4.1 int Client::fClientId [private]

5.1.4.2 bool Client::fIsConnected [private]

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

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

5.2 Exception Class Reference

A simple exception handler.

```
#include <Exception.h>
```

Public Member Functions

- [Exception](#) (const char *from, std::string desc, ExceptionType type=Undefined, const int id=0)
- [Exception](#) (const char *from, const char *desc, ExceptionType type=Undefined, const int id=0)
- [~Exception](#) ()
- std::string [From](#) () const
- int [ErrorNumber](#) () const
- std::string [Description](#) () const
- ExceptionType [Type](#) () const
- std::string [TypeString](#) () const
- void [Dump](#) (std::ostream &os=std::cerr) const

Private Attributes

- std::string [fFrom](#)
- std::string [fDescription](#)
- ExceptionType [fType](#)
- int [fErrorNumber](#)

5.2.1 Detailed Description

A simple exception handler.

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

24 Mar 2015

5.2.2 Constructor & Destructor Documentation

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

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

5.2.2.3 `Exception::~~Exception ()` [inline]

Here is the call graph for this function:

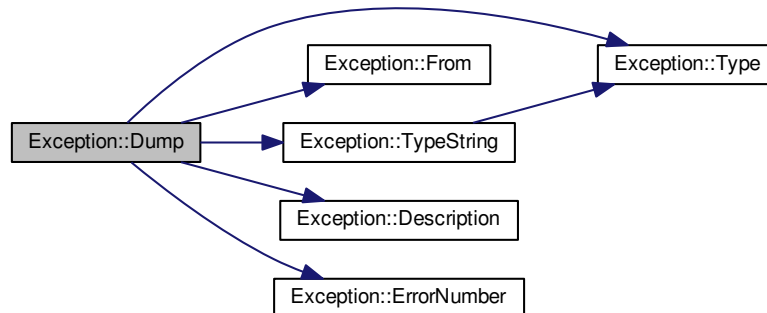


5.2.3 Member Function Documentation

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

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

Here is the call graph for this function:



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

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

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

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

Here is the call graph for this function:



5.2.4 Field Documentation

5.2.4.1 `std::string Exception::fDescription [private]`

5.2.4.2 `int Exception::fErrorNumber [private]`

5.2.4.3 `std::string Exception::fFrom [private]`

5.2.4.4 `ExceptionType Exception::fType [private]`

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

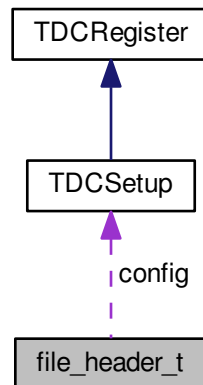
- `include/Exception.h`

5.3 file_header_t Struct Reference

Header to the output files.

```
#include <FPGAHandler.h>
```

Collaboration diagram for file_header_t:



Data Fields

- uint32_t [magic](#)
- uint32_t [run_id](#)
- uint32_t [spill_id](#)
- [TDCSetup config](#)

5.3.1 Detailed Description

Header to the output files.

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

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

14 Apr 2015

5.3.2 Field Documentation

5.3.2.1 TDCSetup file_header_t::config

5.3.2.2 uint32_t file_header_t::magic

5.3.2.3 uint32_t file_header_t::run_id

5.3.2.4 uint32_t file_header_t::spill_id

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

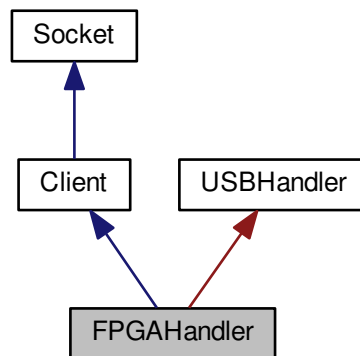
- include/FPGAHandler.h

5.4 FPGAHandler Class Reference

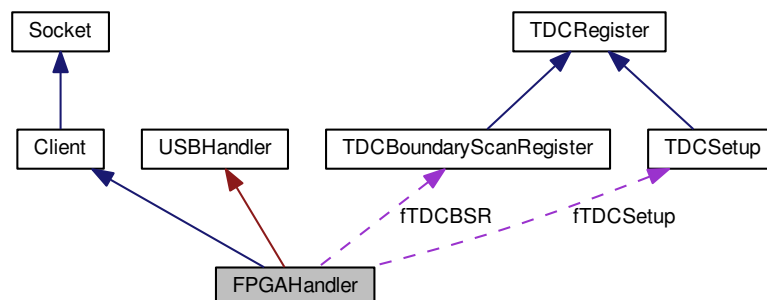
Driver for timing detectors' FPGA readout.

```
#include <FPGAHandler.h>
```

Inheritance diagram for FPGAHandler:



Collaboration diagram for FPGAHandler:



Public Member Functions

- [FPGAHandler](#) (int port, const char *dev)

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

- virtual [~FPGAHandler](#) ()
- void [OpenFile](#) ()

Open an output file to store header/HPTDC events.

- void [CloseFile](#) ()

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

- std::string [GetFilename](#) () const

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

- void [SetConfiguration](#) (const [TDCSetup](#) &c)

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

- [TDCSetup](#) [GetConfiguration](#) ()

Retrieve the HPTDC setup word as a [TDCSetup](#) object.

- void [ReadBuffer](#) ()
- [SocketType](#) [GetType](#) () const

[Socket](#) actor type retrieval method.

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.

- void [ReadBoundaryScanRegister](#) ()

Private Attributes

- std::string [fFilename](#)
- std::ofstream [fOutput](#)
- [TDCSetup](#) [fTDCSetup](#)
- [TDCBoundaryScanRegister](#) [fTDCBSR](#)
- bool [fIsFileOpen](#)
- bool [fIsTDCInReadout](#)

Additional Inherited Members

5.4.1 Detailed Description

Driver for timing detectors' FPGA readout.

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

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

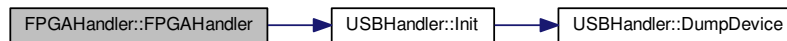
14 Apr 2015

5.4.2 Constructor & Destructor Documentation

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

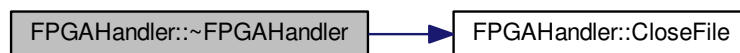
Bind to a FPGA through the USB protocol, and to the socket.

Here is the call graph for this function:



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

Here is the call graph for this function:



5.4.3 Member Function Documentation

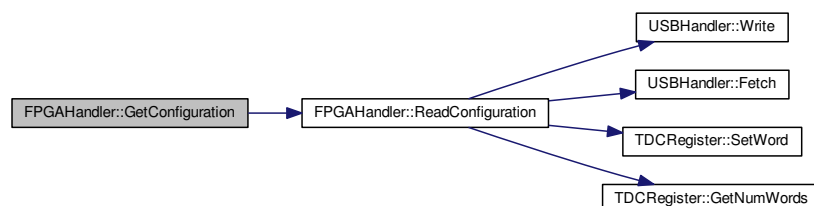
5.4.3.1 void FPGAHandler::CloseFile ()

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

5.4.3.2 TDCSetup FPGAHandler::GetConfiguration () [inline]

Retrieve the HPTDC setup word as a [TDCSetup](#) object.

Here is the call graph for this function:



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

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

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

[Socket](#) actor type retrieval method.

Reimplemented from [Client](#).

5.4.3.5 `void FPGAHandler::OpenFile ()`

Open an output file to store header/HPTDC events.

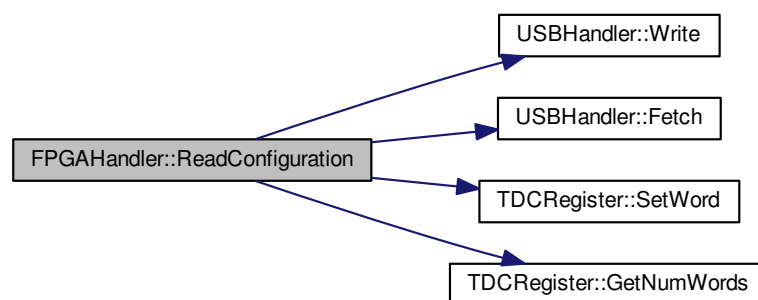
5.4.3.6 `void FPGAHandler::ReadBoundaryScanRegister () [private]`

5.4.3.7 `void FPGAHandler::ReadBuffer ()`

5.4.3.8 `void FPGAHandler::ReadConfiguration () [private]`

Read the setup word from the HPTDC internal setup register.

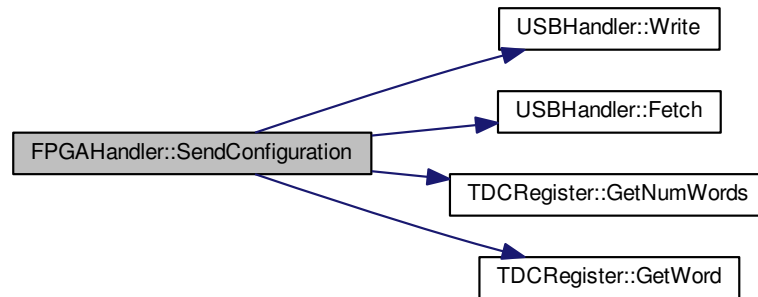
Here is the call graph for this function:



5.4.3.9 `void FPGAHandler::SendConfiguration () [private]`

Set the setup word to the HPTDC internal setup register.

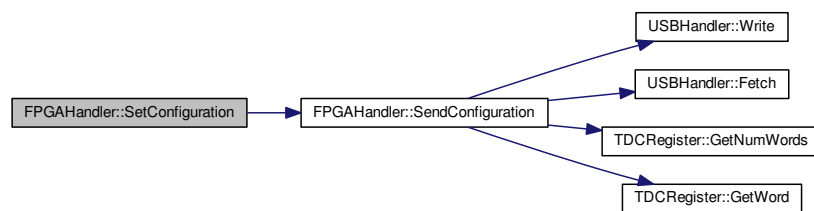
Here is the call graph for this function:



5.4.3.10 `void FPGAHandler::SetConfiguration (const TDCSetup & c) [inline]`

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

Here is the call graph for this function:



5.4.4 Field Documentation

5.4.4.1 `std::string FPGAHandler::fFilename [private]`

5.4.4.2 `bool FPGAHandler::fIsFileOpen [private]`

5.4.4.3 `bool FPGAHandler::fIsTDCInReadout [private]`

5.4.4.4 `std::ofstream FPGAHandler::fOutput [private]`

5.4.4.5 `TDCBoundaryScanRegister FPGAHandler::fTDCBSR [private]`

5.4.4.6 `TDCSetup FPGAHandler::fTDCSetup [private]`

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

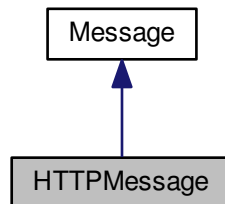
- `include/FPGAHandler.h`
- `src/FPGAHandler.cpp`

5.5 HTTPMessage Class Reference

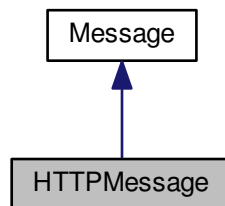
[Message](#) to be transmitted through a WebSocket protocol.

```
#include <HTTPMessage.h>
```

Inheritance diagram for HTTPMessage:



Collaboration diagram for HTTPMessage:



Public Member Functions

- [HTTPMessage](#) (WebSocket *ws, [Message](#) m, MessageAction a)
- [HTTPMessage](#) (WebSocket *ws, const char *msg, MessageAction a)
- void [Decode](#) ()
- void [Encode](#) ()
- MessageKey [GetKey](#) () const
- void [Dump](#) (std::ostream &os=std::cout) const

Private Attributes

- WebSocket * [fWS](#)
- std::string [fOriginalString](#)

Additional Inherited Members

5.5.1 Detailed Description

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

Type of message compatible to the transmission through a WebSocket protocol. It enables a direct conversion of standards from any socket message format used elsewhere in this code using the *MessageAction* statement.

Author

Laurent Forthomme laurent.forthomme@cern.ch

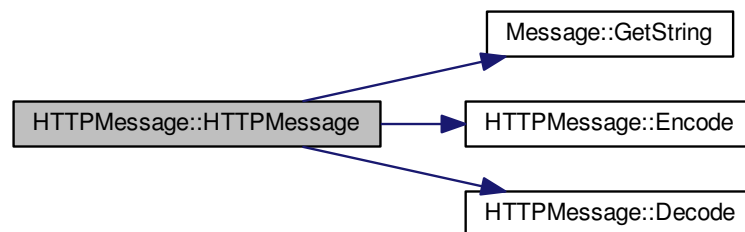
Date

1 Apr 2015

5.5.2 Constructor & Destructor Documentation

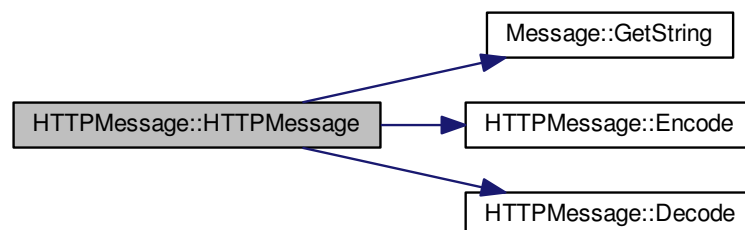
5.5.2.1 HTTPMessage::HTTPMessage (WebSocket * ws, Message m, MessageAction a) [inline]

Here is the call graph for this function:



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

Here is the call graph for this function:



5.5.3 Member Function Documentation

5.5.3.1 void HTTPMessage::Decode () [inline]

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

5.5.3.3 void HTTPMessage::Encode () [inline]

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

5.5.4 Field Documentation

5.5.4.1 std::string HTTPMessage::fOriginalString [private]

5.5.4.2 WebSocket* HTTPMessage::fWS [private]

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

- include/HTTPMessage.h

5.6 ListenerInfo Struct Reference

Information on a socket's listener.

```
#include <Messenger.h>
```

Data Fields

- std::string [name](#)
- [Socket::SocketType](#) type

5.6.1 Detailed Description

Information on a socket's listener.

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

5.6.2 Field Documentation

5.6.2.1 std::string ListenerInfo::name

5.6.2.2 [Socket::SocketType](#) ListenerInfo::type

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

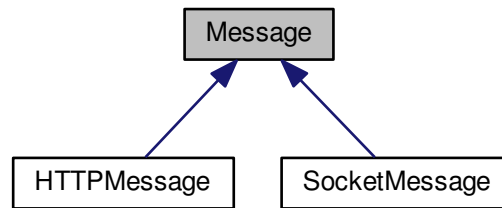
- include/Messenger.h

5.7 Message Class Reference

Base socket message type.

```
#include <Message.h>
```

Inheritance diagram for Message:



Public Member Functions

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

Protected Attributes

- `std::string fString`

5.7.1 Detailed Description

Base socket message type.

Base handler for messages to be transmitted through the socket

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

6 Apr 2015

5.7.2 Constructor & Destructor Documentation

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

Void message constructor.

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

Construct a message from a string.

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

Construct a message from a string.

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

5.7.3 Member Function Documentation

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

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

Placeholder for the MessageKey retrieval method.

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

Retrieve the string carried by this message as a whole.

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

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

5.7.4 Field Documentation

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

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

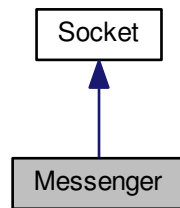
- `include/Message.h`

5.8 Messenger Class Reference

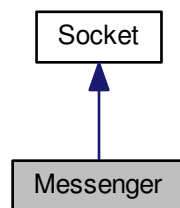
Base master object for the socket.

```
#include <Messenger.h>
```

Inheritance diagram for Messenger:



Collaboration diagram for Messenger:



Public Member Functions

- [Messenger](#) ()
Build a void master object or socket actor.
- [Messenger](#) (int port)
Build a master object to control the socket.
- [~Messenger](#) ()
- bool [Connect](#) ()
Connect the master to the socket.
- void [Disconnect](#) ()
Remove the master and destroy the socket.
- void [Send](#) (const [Message](#) &m, int sid) const
Send any type of message to any client.
- void [Receive](#) ()
Handle a message reception from a client.
- void [Broadcast](#) (const [Message](#) &m) const
Emit a message to all clients connected through the socket.
- [SocketType GetType](#) () const
Socket actor type retrieval method.

Private Member Functions

- void [AddClient](#) ()
Add a client to listen to.
- void [DisconnectClient](#) (int sid, MessageKey key, bool force=false)
Disconnect a client.
- void [SwitchClientType](#) (int sid, [Socket::SocketType](#) type)
- void [ProcessMessage](#) ([SocketMessage](#) m, int sid)
Process a message received from the socket.

Private Attributes

- WebSocket * [fWS](#)
- int [fNumAttempts](#)
- std::vector< [ListenerInfo](#) > [fListenersInfo](#)

Additional Inherited Members

5.8.1 Detailed Description

Base master object for the socket.

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

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

23 Mar 2015

5.8.2 Constructor & Destructor Documentation

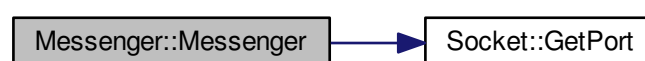
5.8.2.1 [Messenger::Messenger](#) ()

Build a void master object or socket actor.

5.8.2.2 [Messenger::Messenger](#) (int *port*)

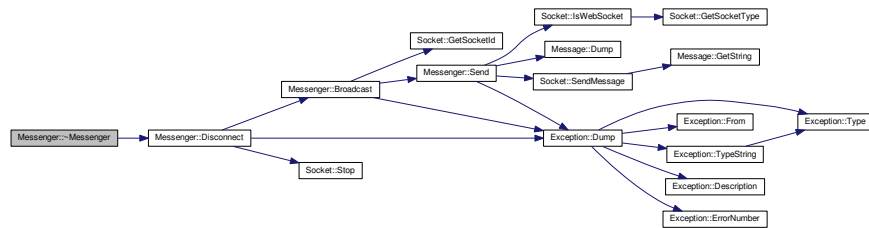
Build a master object to control the socket.

Here is the call graph for this function:



5.8.2.3 Messenger::~~Messenger ()

Here is the call graph for this function:



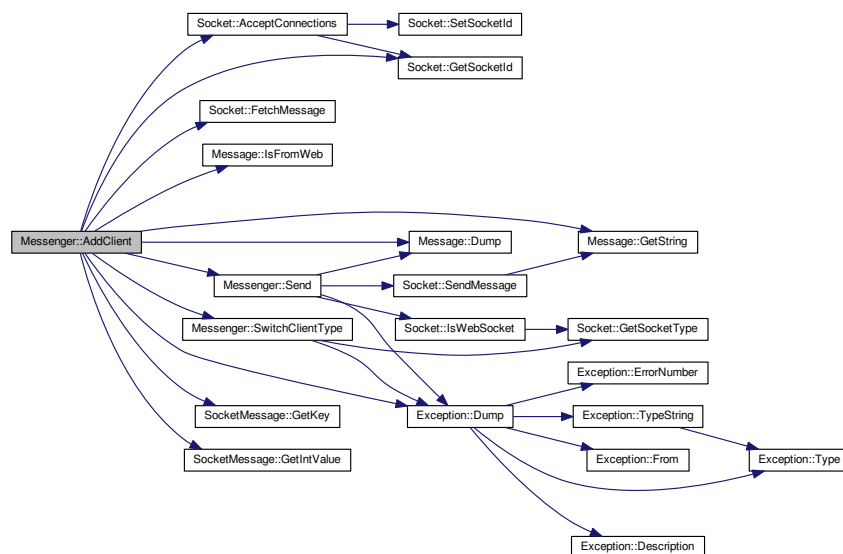
5.8.3 Member Function Documentation

5.8.3.1 void Messenger::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:



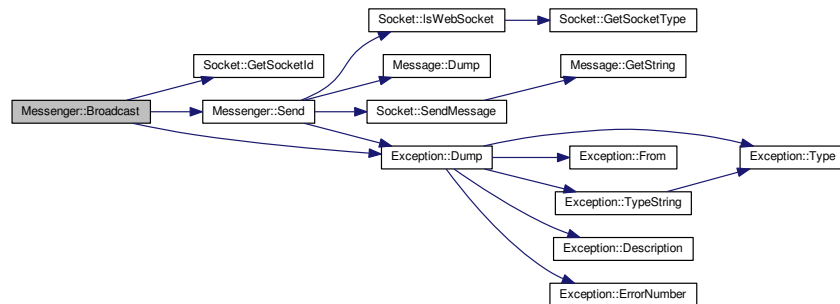
5.8.3.2 void Messenger::Broadcast (const Message & m) const

Emit a message to all clients connected through the socket.

Parameters

in	m	Message to transmit
----	---	---------------------

Here is the call graph for this function:

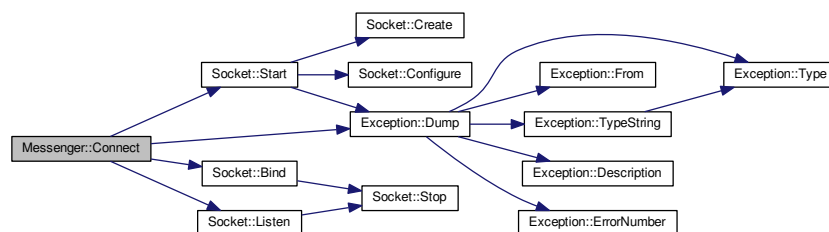


5.8.3.3 bool Messenger::Connect ()

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:

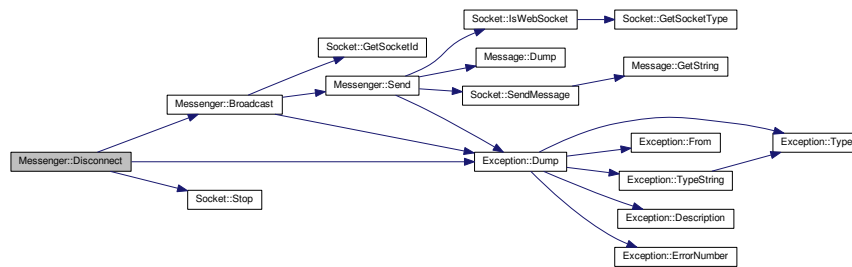


5.8.3.4 void Messenger::Disconnect ()

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:



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

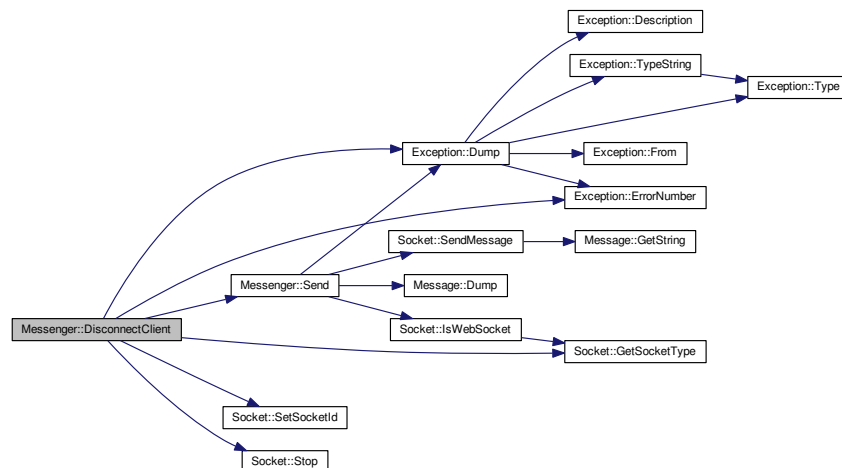
Disconnect a client.

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:



5.8.3.6 `SocketType Messenger::GetType () const` [inline]

[Socket](#) actor type retrieval method.

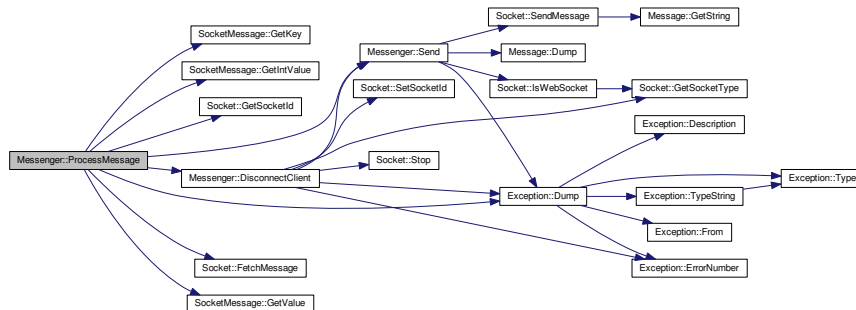
5.8.3.7 `void Messenger::ProcessMessage (SocketMessage m, int sid)` [private]

Process a message received from the socket.

Parameters

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

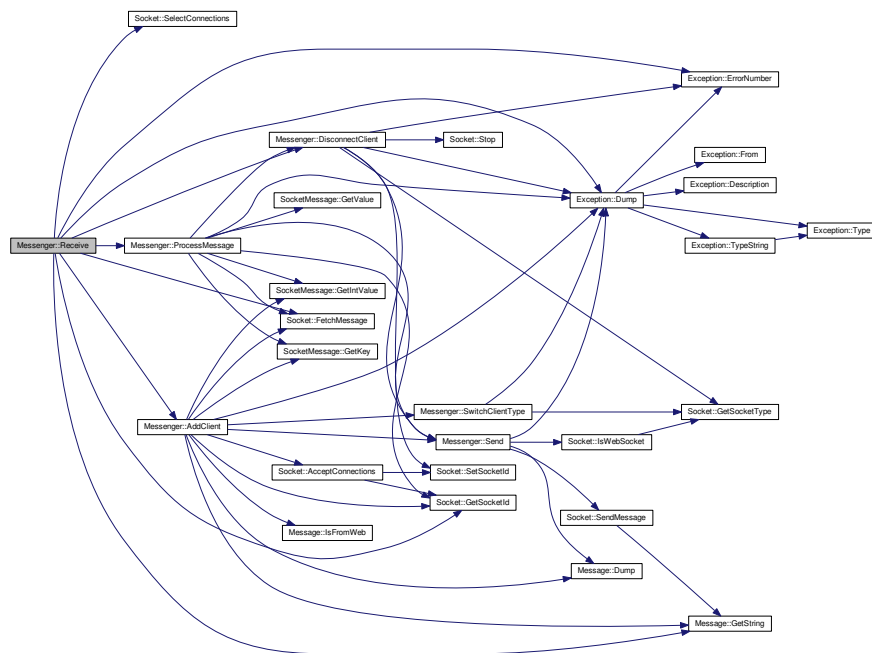
Here is the call graph for this function:



5.8.3.8 void Messenger::Receive ()

Handle a message reception from a client.

Here is the call graph for this function:



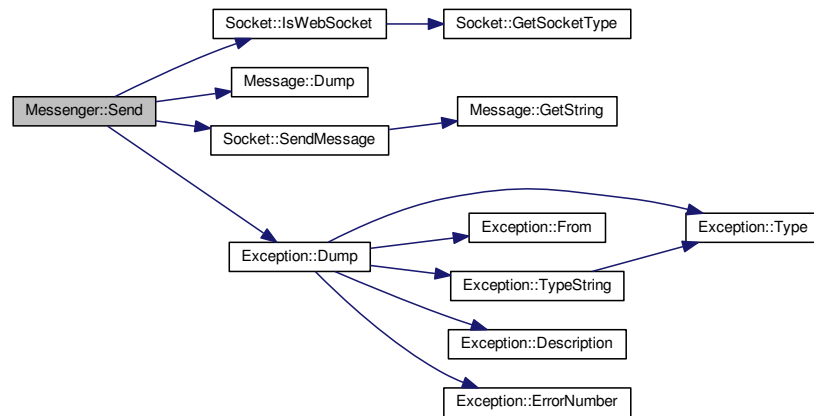
5.8.3.9 void Messenger::Send (const Message & m, int sid) const [inline]

Send any type of message to any client.

Parameters

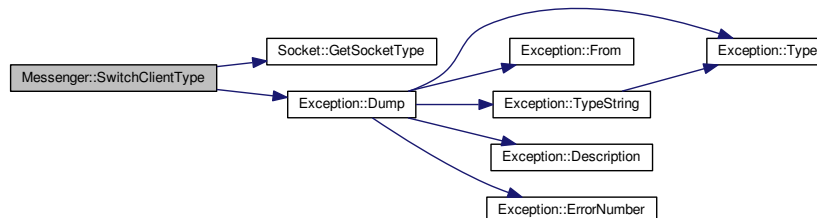
in	<i>m</i>	Message to transmit
in	<i>sid</i>	Unique identifier of the client on this socket

Here is the call graph for this function:



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

Here is the call graph for this function:



5.8.4 Field Documentation

5.8.4.1 std::vector<ListenerInfo> Messenger::fListenersInfo [private]

5.8.4.2 int Messenger::fNumAttempts [private]

5.8.4.3 WebSocket* Messenger::fWS [private]

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

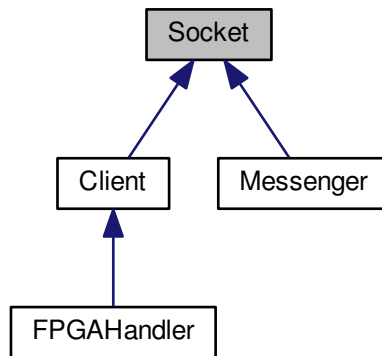
- include/Messenger.h
- src/Messenger.cpp

5.9 Socket Class Reference

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

```
#include <Socket.h>
```

Inheritance diagram for Socket:



Public Types

- enum `SocketType` {
`INVALID` = -1, `MASTER` = 0, `WEBSOCKET_CLIENT`, `CLIENT`,
`DETECTOR` }
- *Type of actor playing a role on the socket.*
- typedef `std::set< std::pair< int, SocketType > >` `SocketCollection`

Public Member Functions

- `Socket ()`
- `Socket (int port)`
- `virtual ~Socket ()`
- `void Stop ()`
Terminates the socket and all attached communications.
- `void SetPort (int port)`
- `int GetPort () const`
Retrieve the port used for this socket.
- `void AcceptConnections (Socket &socket)`
Accept connection from a client.
- `void SelectConnections ()`
- `void SetSocketId (int sid)`
- `int GetSocketId () const`
- `SocketType GetSocketType (int sid) const`
- `bool IsWebSocket (int sid) const`
- `void DumpConnected () const`

Protected Member Functions

- bool [Start](#) ()
Start the socket.
- void [Bind](#) ()
Bind a name to a socket.
- void [PrepareConnection](#) ()
- void [Listen](#) (int maxconn)
Listen to incoming messages.
- void [SendMessage](#) ([Message](#) message, int id=-1) const
Send a message on a socket.
- [Message](#) [FetchMessage](#) (int id=-1) const
Receive a message from a socket.

Protected Attributes

- int [fPort](#)
- char [fBuffer](#) [MAX_WORD_LENGTH]
- [SocketCollection](#) [fSocketsConnected](#)
- fd_set [fMaster](#)
Master file descriptor list.
- fd_set [fReadFds](#)
Temp file descriptor list for select()

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](#)

5.9.1 Detailed Description

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

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

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

23 Mar 2015

5.9.2 Member Typedef Documentation

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

5.9.3 Constructor & Destructor Documentation

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

5.9.3.2 `Socket::Socket (int port)`

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

5.9.4 Member Function Documentation

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

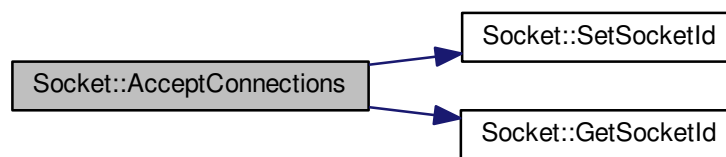
Accept connection from a client.

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

Parameters

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

Here is the call graph for this function:



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

Bind a name to a socket.

Returns

Success of the operation

Here is the call graph for this function:



5.9.4.3 void Socket::Configure () [private]

Configure the socket object for communication.

5.9.4.4 void Socket::Create () [private]

Create an endpoint for communication.

5.9.4.5 void Socket::DumpConnected () const

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

Receive a message from a socket.

Returns

Received message as a std::string

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

Retrieve the port used for this socket.

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

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

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

Here is the call graph for this function:



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

Listen to incoming messages.

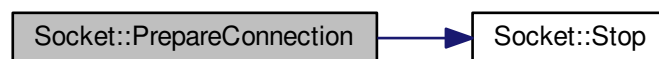
Set the socket to listen to any message coming from outside

Here is the call graph for this function:



5.9.4.12 void Socket::PrepareConnection () [protected]

Here is the call graph for this function:



5.9.4.13 void Socket::SelectConnections ()

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

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

Send a message on a socket.

Here is the call graph for this function:



5.9.4.15 void Socket::SetPort (int port) [inline]

5.9.4.16 void Socket::SetSocketId (int sid) [inline]

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

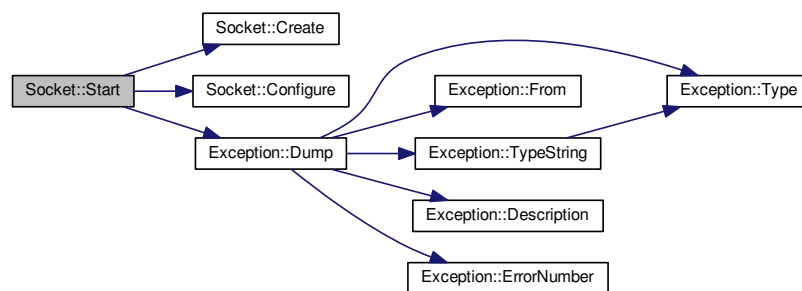
Start the socket.

Launch all mandatory operations to set the socket to be used

Returns

Success of the operation

Here is the call graph for this function:



5.9.4.18 `void Socket::Stop ()`

Terminates the socket and all attached communications.

5.9.5 Field Documentation

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

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

5.9.5.3 `fd_set Socket::fMaster` [protected]

Master file descriptor list.

5.9.5.4 `int Socket::fPort` [protected]

5.9.5.5 `fd_set Socket::fReadFds` [protected]

Temp file descriptor list for select()

5.9.5.6 `int Socket::fSocketId` [private]

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

5.9.5.7 `SocketCollection Socket::fSocketsConnected` [protected]

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

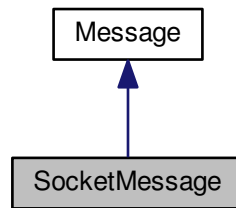
- `include/Socket.h`
- `src/Socket.cpp`

5.10 SocketMessage Class Reference

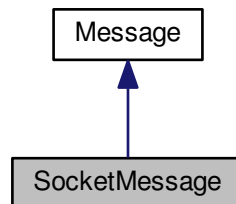
Socket-passed message type.

```
#include <SocketMessage.h>
```

Inheritance diagram for SocketMessage:



Collaboration diagram for SocketMessage:



Public Member Functions

- [SocketMessage](#) ()
- [SocketMessage](#) (const [Message](#) &msg)
- [SocketMessage](#) (const char *msg_s)
- [SocketMessage](#) (std::string msg_s)
- [SocketMessage](#) (const MessageKey &key)
Construct a socket message out of a key.
- [SocketMessage](#) (const MessageKey &key, const char *value)
Construct a socket message out of a key and a string-type value.
- [SocketMessage](#) (const MessageKey &key, std::string value)
Construct a socket message out of a key and a string-type value.

- [SocketMessage](#) (const MessageKey &key, const int value)
Construct a socket message out of a key and an integer-type value.
- [SocketMessage](#) (const MessageKey &key, const float value)
Construct a socket message out of a key and a float-type value.
- [SocketMessage](#) (const MessageKey &key, const double value)
Construct a socket message out of a key and a double precision-type value.
- [SocketMessage](#) (MessageMap msg_m)
Construct a socket message out of a map of key/string-type value.
- [~SocketMessage](#) ()
- void [SetKeyValue](#) (const MessageKey &key, const char *value)
String-valued message.
- void [SetKeyValue](#) (const MessageKey &key, int int_value)
Send an integer-valued message.
- void [SetKeyValue](#) (const MessageKey &key, float float_value)
Float-valued message.
- void [SetKeyValue](#) (const MessageKey &key, double double_value)
Double-valued message.
- std::string [GetString](#) () const
Extract the whole key:value message.
- MessageKey [GetKey](#) () const
Extract the message's key.
- std::string [GetValue](#) () const
Extract the message's string value.
- int [GetIntValue](#) () const
Extract the message's integer value.
- VectorValue [GetVectorValue](#) () const
Extract the message's vector of string value.
- void [Dump](#) (std::ostream &os=std::cout) const

Private Member Functions

- MessageMap [Object](#) () const
- std::string [String](#) () const

Private Attributes

- MessageMap [fMessage](#)

Additional Inherited Members

5.10.1 Detailed Description

Socket-passed message type.

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

26 Mar 2015

5.10.2 Constructor & Destructor Documentation

5.10.2.1 `SocketMessage::SocketMessage ()` `[inline]`

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

Here is the call graph for this function:



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

Here is the call graph for this function:



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

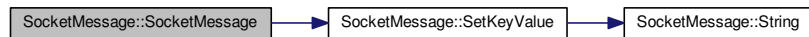
Here is the call graph for this function:



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

Construct a socket message out of a key.

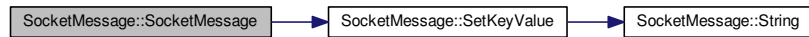
Here is the call graph for this function:



5.10.2.6 `SocketMessage::SocketMessage (const MessageKey & key, const char * value) [inline]`

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

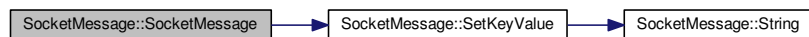
Here is the call graph for this function:



5.10.2.7 `SocketMessage::SocketMessage (const MessageKey & key, std::string value) [inline]`

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

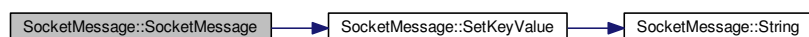
Here is the call graph for this function:



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

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

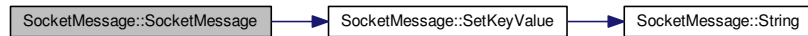
Here is the call graph for this function:



5.10.2.9 `SocketMessage::SocketMessage (const MessageKey & key, const float value) [inline]`

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

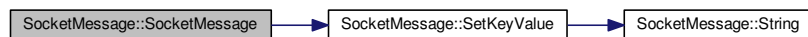
Here is the call graph for this function:



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

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

Here is the call graph for this function:



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

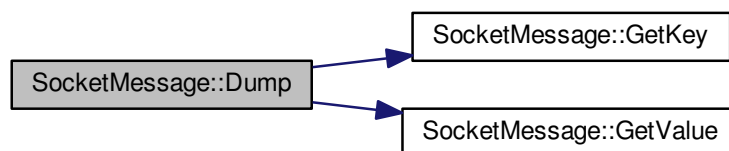
Construct a socket message out of a map of key/string-type value.

5.10.2.12 SocketMessage::~~SocketMessage () [inline]

5.10.3 Member Function Documentation

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

Here is the call graph for this function:



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

Extract the message's integer value.

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

Extract the message's key.

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

Extract the whole key:value message.

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

Extract the message's string value.

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

Extract the message's vector of string value.

Here is the call graph for this function:



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

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

String-valued message.

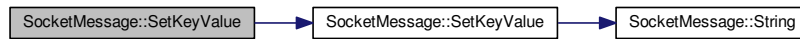
Here is the call graph for this function:



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

Send an integer-valued message.

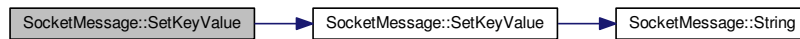
Here is the call graph for this function:



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

Float-valued message.

Here is the call graph for this function:



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

Double-valued message.

Here is the call graph for this function:



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

5.10.4 Field Documentation

5.10.4.1 `MessageMap SocketMessage::fMessage [private]`

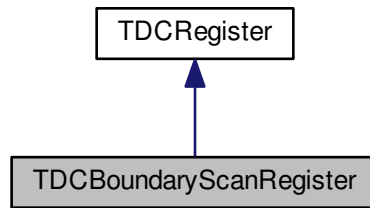
The documentation for this class was generated from the following file:

- `include/SocketMessage.h`

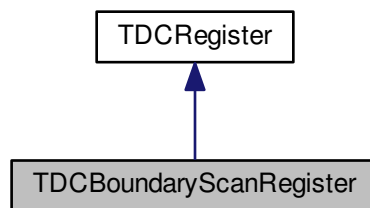
5.11 TDCBoundaryScanRegister Class Reference

```
#include <TDCBoundaryScan.h>
```

Inheritance diagram for TDCBoundaryScanRegister:



Collaboration diagram for TDCBoundaryScanRegister:



Public Member Functions

- [TDCBoundaryScanRegister\(\)](#)
- virtual [~TDCBoundaryScanRegister\(\)](#)

Additional Inherited Members

5.11.1 Detailed Description

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

24 Apr 2015

5.11.2 Constructor & Destructor Documentation

5.11.2.1 TDCBoundaryScanRegister::TDCBoundaryScanRegister() [inline]

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

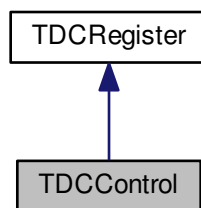
The documentation for this class was generated from the following file:

- `include/TDCBoundaryScan.h`

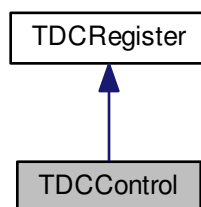
5.12 TDCControl Class Reference

```
#include <TDCControl.h>
```

Inheritance diagram for TDCControl:



Collaboration diagram for TDCControl:



Public Types

- enum [EnablePattern](#)

Public Member Functions

- [TDCControl](#) ()
- [TDCControl](#) (const [TDCControl](#) &c)
- virtual [~TDCControl](#) ()
- void [SetEnablePattern](#) (const [EnablePattern](#) &ep)
- void [SetGlobalReset](#) (const bool gr=true)

- void [SetDLLReset](#) (const bool dr=true)
- void [SetPLLReset](#) (const bool pr=true)
- void [SetControlParity](#) (const bool cp=true)
- void [Dump](#) (int verb=1, std::ostream &os=std::cout) const

Static Private Attributes

- static const [bit kEnablePattern](#) = 0
- static const [bit kGlobalReset](#) = 4
- static const [bit kEnableChannel](#) = 5
- static const [bit kDLLReset](#) = 37
- static const [bit kPLLReset](#) = 38
- static const [bit kControlParity](#) = 39

Additional Inherited Members

5.12.1 Member Enumeration Documentation

5.12.1.1 enum [TDCControl::EnablePattern](#)

5.12.2 Constructor & Destructor Documentation

5.12.2.1 [TDCControl::TDCControl \(\)](#) [inline]

5.12.2.2 [TDCControl::TDCControl \(const \[TDCControl\]\(#\) & c \)](#) [inline]

Here is the call graph for this function:



5.12.2.3 virtual [TDCControl::~~TDCControl \(\)](#) [inline],[virtual]

5.12.3 Member Function Documentation

5.12.3.1 void [TDCControl::Dump \(int verb = 1, std::ostream & os = std::cout \) const](#) [inline]

Here is the call graph for this function:



5.12.3.2 `void TDCCControl::SetControlParity (const bool cp = true) [inline]`

Here is the call graph for this function:



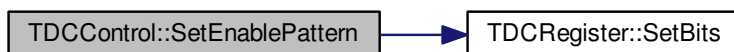
5.12.3.3 `void TDCCControl::SetDLLReset (const bool dr = true) [inline]`

Here is the call graph for this function:



5.12.3.4 `void TDCCControl::SetEnablePattern (const EnablePattern & ep) [inline]`

Here is the call graph for this function:



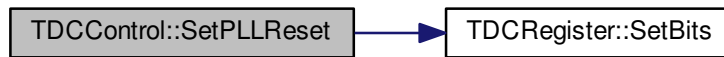
5.12.3.5 `void TDCCControl::SetGlobalReset (const bool gr = true) [inline]`

Here is the call graph for this function:



5.12.3.6 `void TDCControl::SetPLLReset (const bool pr = true) [inline]`

Here is the call graph for this function:



5.12.4 Field Documentation

5.12.4.1 `const bit TDCControl::kControlParity = 39 [static], [private]`

5.12.4.2 `const bit TDCControl::kDLLReset = 37 [static], [private]`

5.12.4.3 `const bit TDCControl::kEnableChannel = 5 [static], [private]`

5.12.4.4 `const bit TDCControl::kEnablePattern = 0 [static], [private]`

5.12.4.5 `const bit TDCControl::kGlobalReset = 4 [static], [private]`

5.12.4.6 `const bit TDCControl::kPLLReset = 38 [static], [private]`

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

- `include/TDCControl.h`

5.13 TDCEvent Class Reference

HPTDC event parser.

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

Public Types

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

Public Member Functions

- `TDCEvent` (const uint32_t &word)
- virtual `~TDCEvent` ()
- `EventType GetType` () const
Type of packet read out from the TDC.
- unsigned int `GetTDCId` () const
Programmed identifier of master TDC.

- uint16_t [GetEventId](#) () const
Event identifier from event counter.
- uint16_t [GetWordCount](#) () const
Total number of words in event (including headers and trailers)
- uint16_t [GetBunchId](#) () const
Bunch identifier of trigger (or trigger time tag)
- uint32_t [GetLeadingTime](#) (bool pair=false) const
Leading edge measurement in programmed time resolution.
- uint8_t [GetWidth](#) () const
Width of pulse in programmed time resolution.
- uint32_t [GetTrailingTime](#) () const
Trailing edge measurement in programmed time resolution.
- uint16_t [GetErrorFlags](#) () const
Return error flags if an error condition has been detected.

Private Attributes

- uint32_t [fWord](#)

5.13.1 Detailed Description

HPTDC event parser.

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

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

20 Apr 2015

5.13.2 Constructor & Destructor Documentation

5.13.2.1 TDCEvent::TDCEvent (const uint32_t & word) [inline]

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

5.13.3 Member Function Documentation

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

Bunch identifier of trigger (or trigger time tag)

Here is the call graph for this function:



5.13.3.2 `uint16_t TDCEvent::GetErrorFlags () const [inline]`

Return error flags if an error condition has been detected.

Here is the call graph for this function:



5.13.3.3 `uint16_t TDCEvent::GetEventId () const [inline]`

Event identifier from event counter.

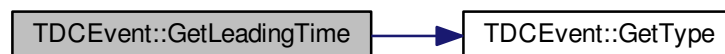
Here is the call graph for this function:



5.13.3.4 `uint32_t TDCEvent::GetLeadingTime (bool pair = false) const [inline]`

Leading edge measurement in programmed time resolution.

Here is the call graph for this function:



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

Programmed identifier of master TDC.

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

Trailing edge measurement in programmed time resolution.

Here is the call graph for this function:



5.13.3.7 `EventType TDCEvent::GetType () const [inline]`

Type of packet read out from the TDC.

5.13.3.8 `uint8_t TDCEvent::GetWidth () const [inline]`

Width of pulse in programmed time resolution.

Here is the call graph for this function:



5.13.3.9 `uint16_t TDCEvent::GetWordCount () const [inline]`

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

Here is the call graph for this function:



5.13.4 Field Documentation

5.13.4.1 uint32_t TDCEvent::fWord [private]

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

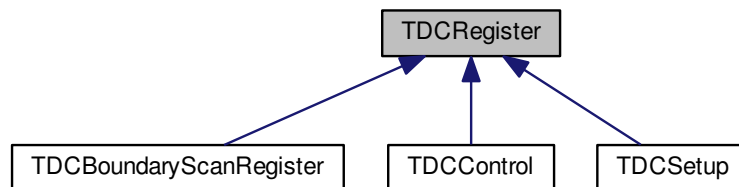
- include/TDCEvent.h

5.14 TDCRegister Class Reference

General register object to interact with a HPTDC chip.

```
#include <TDCRegister.h>
```

Inheritance diagram for TDCRegister:



Public Types

- typedef uint16_t **bit**
LSB index.
- typedef uint32_t **word_t**
Unit of the TDC register word to be successfully contained on any machine.

Public Member Functions

- **TDCRegister** (const size_t size)
- virtual **~TDCRegister** ()
- void **SetWord** (const unsigned int i, const **word_t** word)
Set one bit(s) subset in the register word.
- **word_t** **GetWord** (const unsigned int i) const
Retrieve one subset from the register word.
- uint8_t **GetNumWords** () const
Number of words in the register.
- void **DumpRegister** (**bit** max_bits=-1, std::ostream &os=std::cout) const

Protected Member Functions

- void **SetBits** (uint16_t lsb, uint16_t word, uint8_t size)
Set bits in the register word.
- uint16_t **GetBits** (uint16_t lsb, uint8_t size) const
Extract bits from the register word.

Protected Attributes

- `word_t * fWord`
- `size_t fNumWords`

5.14.1 Detailed Description

General register object to interact with a HPTDC chip.

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

24 Apr 2015

5.14.2 Member Typedef Documentation

5.14.2.1 `typedef uint16_t TDCRegister::bit`

LSB index.

5.14.2.2 `typedef uint32_t TDCRegister::word_t`

Unit of the TDC register word to be successfully contained on any machine.

5.14.3 Constructor & Destructor Documentation

5.14.3.1 `TDCRegister::TDCRegister (const size_t size) [inline]`

5.14.3.2 `virtual TDCRegister::~~TDCRegister () [inline], [virtual]`

5.14.4 Member Function Documentation

5.14.4.1 `void TDCRegister::DumpRegister (bit max_bits = -1, std::ostream & os = std::cout) const [inline]`

5.14.4.2 `uint16_t TDCRegister::GetBits (uint16_t lsb, uint8_t size) const [inline], [protected]`

Extract bits from the register word.

Extract a fixed amount of bits from the full register word

Parameters

<code>in</code>	<code>lsb</code>	Least significant bit of the word to retrieve
<code>in</code>	<code>size</code>	Size of the word to retrieve

5.14.4.3 `uint8_t TDCRegister::GetNumWords () const [inline]`

Number of words in the register.

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

5.14.4.4 `word_t TDCRegister::GetWord (const unsigned int i) const` `[inline]`

Retrieve one subset from the register word.

5.14.4.5 `void TDCRegister::SetBits (uint16_t lsb, uint16_t word, uint8_t size)` `[inline]`, `[protected]`

Set bits in the register word.

Set a fixed amount of bits in the full register word

Parameters

in	<i>lsb</i>	Least significant bit of the word to set
in	<i>word</i>	Word to set
in	<i>size</i>	Size of the word to set

5.14.4.6 `void TDCRegister::SetWord (const unsigned int i, const word_t word)` `[inline]`

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

5.14.5 Field Documentation

5.14.5.1 `size_t TDCRegister::fNumWords` `[protected]`

5.14.5.2 `word_t* TDCRegister::fWord` `[protected]`

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

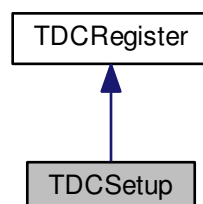
- include/TDCRegister.h

5.15 TDCSetup Class Reference

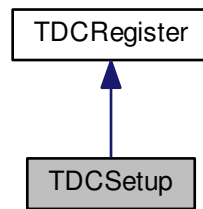
Setup word to be sent to the HPTDC chip.

```
#include <TDCSetup.h>
```

Inheritance diagram for TDCSetup:



Collaboration diagram for TDCSetup:



Public Types

- enum [EdgeResolution](#) {
[E_100ps](#) =0, [E_200ps](#), [E_400ps](#), [E_800ps](#),
[E_1p6ns](#), [E_3p12ns](#), [E_6p25ns](#), [E_12p5ns](#) }
- enum [DeadTime](#) { [DT_5ns](#) =0, [DT_10ns](#), [DT_30ns](#), [DT_100ns](#) }
- enum [WidthResolution](#) {
[W_100ps](#) =0, [W_200ps](#), [W_400ps](#), [W_800ps](#),
[W_1p6ns](#), [W_3p2ns](#), [W_6p25ns](#), [W_12p5ns](#),
[W_25ns](#), [W_50ns](#), [W_100ns](#), [W_200ns](#),
[W_400ns](#), [W_800ns](#) }
- enum [EnabledError](#) {
[VernierError](#) =0x1, [CoarseError](#) =0x2, [ChannelSelectError](#) =0x4, [L1BufferParityError](#) =0x8,
[TriggerFIFOParityError](#) =0x10, [TriggerMatchingError](#) =0x20, [ReadoutFIFOParityError](#) =0x40, [ReadoutStateError](#) =0x80,
[SetupParityError](#) =0x100, [ControlParityError](#) =0x200, [JTAGInstructionParityError](#) =0x400 }
- enum [DLLSpeedMode](#) { [DLL_40MHz](#) =0x0, [DLL_160MHz](#) =0x1, [DLL_320MHz](#) =0x2, [DLL_Illegal](#) =0x3 }
- enum [SerialClockSource](#) { [Serial_pll_clock_80](#) =0x0, [Serial_pll_clock_160](#) =0x1, [Serial_pll_clock_40](#) =0x2, [Serial_aux_clock](#) =0x3 }
- enum [IOClockSource](#) { [IO_clock_40](#) =0x0, [IO_pll_clock_80](#) =0x1, [IO_pll_clock_160](#) =0x2, [IO_aux_clock](#) =0x3 }
- enum [CoreClockSource](#) { [Core_clock_40](#) =0x0, [Core_pll_clock_80](#) =0x1, [Core_pll_clock_160](#) =0x2, [Core_aux_clock](#) =0x3 }
- enum [DLLClockSource](#) {
[DLL_clock_40](#) =0x0, [DLL_pll_clock_40](#) =0x1, [DLL_pll_clock_160](#) =0x2, [DLL_pll_clock_320](#) =0x3,
[DLL_aux_clock](#) =0x4 }
- enum [ReadoutSpeed](#) { [RO_Fixed](#) =0x0, [RO_pll_80Mbits_s](#) =0x1 }
- enum [SerialStrobeType](#) { [SS_NoStrobe](#) =0x0, [SS_DSSStrobe](#) =0x1, [SS_LeadingTrailingStrobe](#) =0x2, [SS_LeadingEdge](#) =0x3 }
- enum [ReadoutSingleCycleSpeed](#) {
[RSC_40Mbits_s](#) =0x0, [RSC_20Mbits_s](#) =0x1, [RSC_10Mbits_s](#) =0x2, [RSC_5Mbits_s](#) =0x3,
[RSC_2p5Mbits_s](#) =0x4, [RSC_1p25Mbits_s](#) =0x5, [RSC_625kbits_s](#) =0x6, [RSC_312p5kbits_s](#) =0x7 }

Public Member Functions

- [TDCSetup](#) ()
- [TDCSetup](#) (const [TDCSetup](#) &c)
- virtual [~TDCSetup](#) ()
- void [SetEnableErrorMark](#) (const bool em)

- void [SetChannelOffset](#) (int channel, uint16_t offset)
Set the time offset for one single channel.
- uint16_t [GetChannelOffset](#) (int channel) const
Return the offset for one single channel.
- void [SetAllChannelsOffset](#) (uint16_t offset)
Set the time offset for all channels.
- void [SetCoarseCountOffset](#) (uint16_t cco)
Set offset for the coarse time counter.
- uint16_t [GetCoarseCountOffset](#) () const
Extract offset for the coarse time counter.
- void [SetDLLAdjustment](#) (int tap, uint8_t adj)
Set the DLL taps adjustments with a resolution of ~ 10 ps.
- uint8_t [GetDLLAdjustment](#) (int tap) const
Set the adjustment of DLL taps.
- void [SetAllTapsDLLAdjustment](#) (uint8_t adj)
Extract the adjustment of DLL taps.
- void [SetRCAdjustment](#) (int tap, uint8_t adj)
Set the adjustment of the RC delay line.
- uint8_t [GetRCAdjustment](#) (int tap)
Extract the adjustment of the RC delay line.
- void [SetWidthResolution](#) (const [WidthResolution](#) r)
Set the pulse width resolution when paired measurements are performed.
- [WidthResolution](#) [GetWidthResolution](#) () const
Extract the pulse width resolution when paired measurements are performed.
- void [SetVernierOffset](#) (const uint8_t vo)
Set the offset in vernier decoding.
- uint8_t [GetVernierOffset](#) () const
Extract the offset in vernier decoding.
- void [SetDeadTime](#) (const [DeadTime](#) dt)
Channel dead time between hits.
- [DeadTime](#) [GetDeadTime](#) () const
- void [SetTestInvert](#) (const bool ti=true)
Automatic inversion of test pattern. Only used during production testing.
- bool [GetTestInvert](#) () const
- void [SetTestMode](#) (const bool tm=true)
Test mode where hit data are taken from coretest. Only used during production testing.
- bool [GetTestMode](#) () const
- void [SetTrailingMode](#) (const bool trail=true)
Enable/disable the detection of trailing edges.
- bool [GetTrailingMode](#) () const
Extract the status for the detection of trailing edges.
- void [SetLeadingMode](#) (const bool lead=true)
Enable the detection of leading edges.
- bool [GetLeadingMode](#) () const
Extract the status for the detection of leading edges.
- void [SetTriggerMatchingMode](#) (const bool trig=true)
Set the enable status of trigger matching mode.
- bool [GetTriggerMatchingMode](#) () const
Extract the enable status of trigger matching mode.
- void [SetEdgesPairing](#) (const bool pair=true)
Enable the pairing of leading and trailing edges (overrides individual enable of leading/trailing edges)

- bool [GetEdgesPairing](#) () const
- void [SetSetupParity](#) (const bool sp=true)
Set the parity of setup data (should be an even parity)
- bool [GetSetupParity](#) () const
Extract the parity of setup data (should be an even parity)
- void [SetConstantValues](#) ()
Ensure that the critical constant values are properly set in the setup word.
- uint16_t [GetTriggerLatency](#) () const
Effective trigger latency in number of clock cycles (when no counter roll-over is used)
- void [Dump](#) (int verb=1, std::ostream &os=std::cout) const

Private Member Functions

- void [SetReadoutSingleCycleSpeed](#) (const [ReadoutSingleCycleSpeed](#) rscs=[RSC_40Mbits_s](#))
Serial transmission speed in single cycle mode.
- void [SetSerialDelay](#) (const uint8_t sd=0x0)
Programmable delay of serial input, in time unit ~ 1 ns.
- void [SetStrobeSelect](#) (const [SerialStrobeType](#) ss=[SS_NoStrobe](#))
- void [SetReadoutSpeedSelect](#) (const [ReadoutSpeed](#) rss=[RO_Fixed](#))
Selection of serial read-out speed.
- void [SetTokenDelay](#) (const uint8_t td=0x0)
Programmable delay of token input, in time unit ~ 1 ns.
- void [SetEnableLocalTrailer](#) (const bool elt=true)
Enable of local trailers in read-out.
- void [SetEnableLocalHeader](#) (const bool elh=true)
Enable of local headers in read-out.
- void [SetEnableGlobalTrailer](#) (const bool egt=true)
Enable of global trailers in read-out (only valid for master TDC)
- void [SetEnableGlobalHeader](#) (const bool egh=true)
Enable of global headers in read-out (only valid for master TDC)
- void [SetKeepToken](#) (const bool kt=true)
- void [SetMaster](#) (const bool m=true)
- void [SetEnableBytewise](#) (const bool seb=true)
- void [SetBypassInputs](#) (const bool sbi=true)
Select serial in and token in from bypass inputs.
- void [SetEnableOverflowDetect](#) (const bool eod=true)
Enable overflow detection of L1 buffers (should always be enabled!)
- void [SetEnableRelative](#) (const bool er=true)
- void [SetEnableAutomaticReject](#) (const bool ear=true)
Enable of automatic rejection (should always be enabled if trigger matching mode!)
- void [SetEnableSetCountersOnBunchReset](#) (const bool escobr=true)
Enable all counters to be set on bunch count reset.
- void [SetEnableMasterResetCode](#) (const bool emrc=true)
Enable master reset code on encoded_control.
- void [SetEnableMasterResetOnEventReset](#) (const bool emroer=true)
Enable master reset of whole TDC on event reset.
- void [SetEnableResetChannelBufferWhenSeparator](#) (const bool ercbws=true)
Enable reset channel buffers when separator.
- void [SetEnableSeparatorOnEventReset](#) (const bool esoer=true)
Enable generation of separator on event reset.
- void [SetEnableSeparatorOnBunchReset](#) (const bool esobr=true)

- Enable generation of separator on bunch reset.*

 - void [SetEnableDirectEventReset](#) (const bool eder=true)

Enable of direct event reset input pin (1), otherwise taken from encoded control.
- void [SetEnableDirectBunchReset](#) (const bool edbr=true)

Enable of direct bunch reset input pin (1), otherwise taken from encoded control.
- void [SetEnableDirectTrigger](#) (const bool edt=true)

Enable of direct trigger input pin.
- void [SetLowPowerMode](#) (const bool lpm=true)

Low power mode of channel buffers.
- void [SetDLLControl](#) (const uint8_t dc)

Control of DLL (DLL charge pump levels)
- void [SetModeRCCompression](#) (const bool mrc=true)

Perform RC interpolation on-chip (only valid in very high resolution mode)
- void [SetModeRC](#) (const bool mr=true)

Enable of RR delay lines mode (in very high resolution mode) ; only for channels 0-4-8-12-16-20-24-28 active.
- void [SetDLLMode](#) (const [DLLSpeedMode](#) dsm)

Selection of DLL speed mode.
- void [SetPLLControl](#) (const uint8_t charge_pump_current=0x4, const bool power_down_mode=false, const bool enable_test_outputs=false, const bool invert_connection_to_status=false)

Control of PLL.
- void [SetSerialClockDelay](#) (const bool delay_clock, const uint8_t delay)

Delay of internal serial clock.
- void [SetIOClockDelay](#) (const bool delay_clock, const uint8_t delay)

Delay of internal I/O clock.
- void [SetCoreClockDelay](#) (const bool delay_clock, const uint8_t delay)

Delay of internal core clock.
- void [SetDLLClockDelay](#) (const bool delay_clock, const uint8_t delay)

Delay of internal DLL clock.
- void [SetSerialClockSource](#) (const [SerialClockSource](#) scs)

Selection of source for serial clock.
- void [SetIOClockSource](#) (const [IOClockSource](#) ics)

Selection of clock source for I/O signals.
- void [SetCoreClockSource](#) (const [CoreClockSource](#) ccs)

Selection of clock source for internal logic.
- void [SetDLLClockSource](#) (const [DLLClockSource](#) dcs)

Selection of clock source for DLL.
- void [SetRollOver](#) (const uint16_t ro=0xFFFF)

Counter roll over value, defining maximal count value from where counters will be reset to 0.
- void [SetEnableTTLSerial](#) (const bool ts=true)

Enable LV TTL inputs on serial registers, and disable their drivers.
- void [SetEnableTTLControl](#) (const bool tc=true)

Enable LV TTL inputs on control registers.
- void [SetEnableTTLReset](#) (const bool tr=true)

Enable LV TTL input on reset, otherwise uses LVDS input levels.
- void [SetEnableTTLClock](#) (const bool tc=true)

Enable LV TTL inputs on: clk, aux_clock, otherwise uses LVDS input levels.
- void [SetEnableTTLHit](#) (const bool th=true)

Enable LV TTL input on hit[31:0], otherwise uses LVDS input levels.

Static Private Attributes

- static const [bit kTestSelect](#) = 0
- static const [bit kEnableErrorMark](#) = 4
- static const [bit kEnableErrorBypass](#) = 5
- static const [bit kEnableError](#) = 6
- static const [bit kReadoutSingleCycleSpeed](#) = 17
- static const [bit kSerialDelay](#) = 20
- static const [bit kStrobeSelect](#) = 24
- static const [bit kReadoutSpeedSelect](#) = 26
- static const [bit kTokenDelay](#) = 27
- static const [bit kEnableLocalTrailer](#) = 31
- static const [bit kEnableLocalHeader](#) = 32
- static const [bit kEnableGlobalTrailer](#) = 33
- static const [bit kEnableGlobalHeader](#) = 34
- static const [bit kKeepToken](#) = 35
- static const [bit kMaster](#) = 36
- static const [bit kEnableBytewise](#) = 37
- static const [bit kEnableSerial](#) = 38
- static const [bit kEnableJTAGReadout](#) = 39
- static const [bit kTDCId](#) = 40
- static const [bit kSelectBypassInputs](#) = 44
- static const [bit kReadoutFIFOSize](#) = 45
- static const [bit kRejectCountOffset](#) = 48
- static const [bit kSearchWindow](#) = 60
- static const [bit kMatchWindow](#) = 72
- static const [bit kLeadingResolution](#) = 84
- static const [bit kMaxEventSize](#) = 116
- static const [bit kRejectFIFOFull](#) = 120
- static const [bit kEnableReadoutOccupancy](#) = 121
- static const [bit kEnableReadoutSeparator](#) = 122
- static const [bit kEnableOverflowDetect](#) = 123
- static const [bit kEnableRelative](#) = 124
- static const [bit kEnableAutomaticReject](#) = 125
- static const [bit kEventCountOffset](#) = 126
- static const [bit kTriggerCountOffset](#) = 138
- static const [bit kEnableSetCountersOnBunchReset](#) = 150
- static const [bit kEnableMasterResetCode](#) = 151
- static const [bit kEnableMasterResetOnEventReset](#) = 152
- static const [bit kEnableResetChannelBufferWhenSeparator](#) = 153
- static const [bit kEnableSeparatorOnEventReset](#) = 154
- static const [bit kEnableSeparatorOnBunchReset](#) = 155
- static const [bit kEnableDirectEventReset](#) = 156
- static const [bit kEnableDirectBunchReset](#) = 157
- static const [bit kEnableDirectTrigger](#) = 158
- static const [bit kOffset0](#) = 438
- static const [bit kCoarseCountOffset](#) = 447
- static const [bit kDLLTapAdjust0](#) = 459
- static const [bit kRCAdjust0](#) = 555
- static const [bit kLowPowerMode](#) = 570
- static const [bit kWidthSelect](#) = 571
- static const [bit kVernierOffset](#) = 575
- static const [bit kDLLControl](#) = 580
- static const [bit kDeadTime](#) = 584
- static const [bit kTestInvert](#) = 586

- static const [bit kTestMode](#) = 587
- static const [bit kTrailing](#) = 588
- static const [bit kLeading](#) = 589
- static const [bit kModeRCCompression](#) = 590
- static const [bit kModeRC](#) = 591
- static const [bit kDLLMode](#) = 592
- static const [bit kPLLControl](#) = 594
- static const [bit kSerialClockDelay](#) = 602
- static const [bit kIOClockDelay](#) = 606
- static const [bit kCoreClockDelay](#) = 610
- static const [bit kDLLClockDelay](#) = 614
- static const [bit kSerialClockSource](#) = 618
- static const [bit kIOClockSource](#) = 620
- static const [bit kCoreClockSource](#) = 622
- static const [bit kDLLClockSource](#) = 624
- static const [bit kRollOver](#) = 627
- static const [bit kEnableMatching](#) = 639
- static const [bit kEnablePair](#) = 640
- static const [bit kEnableTTLSerial](#) = 641
- static const [bit kEnableTTLControl](#) = 642
- static const [bit kEnableTTLReset](#) = 643
- static const [bit kEnableTTLClock](#) = 644
- static const [bit kEnableTTLHit](#) = 645
- static const [bit kSetupParity](#) = 646

Additional Inherited Members

5.15.1 Detailed Description

Setup word to be sent to the HPTDC chip.

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

Author

Laurent Forthomme laurent.forthomme@cern.ch

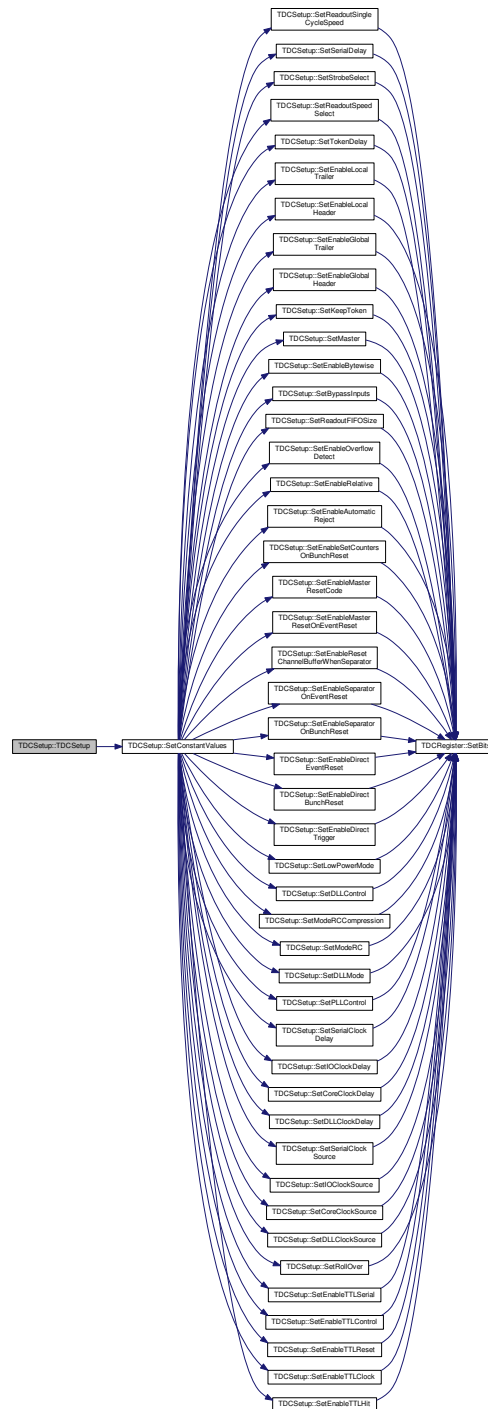
Date

16 Apr 2015

5.15.2 Constructor & Destructor Documentation

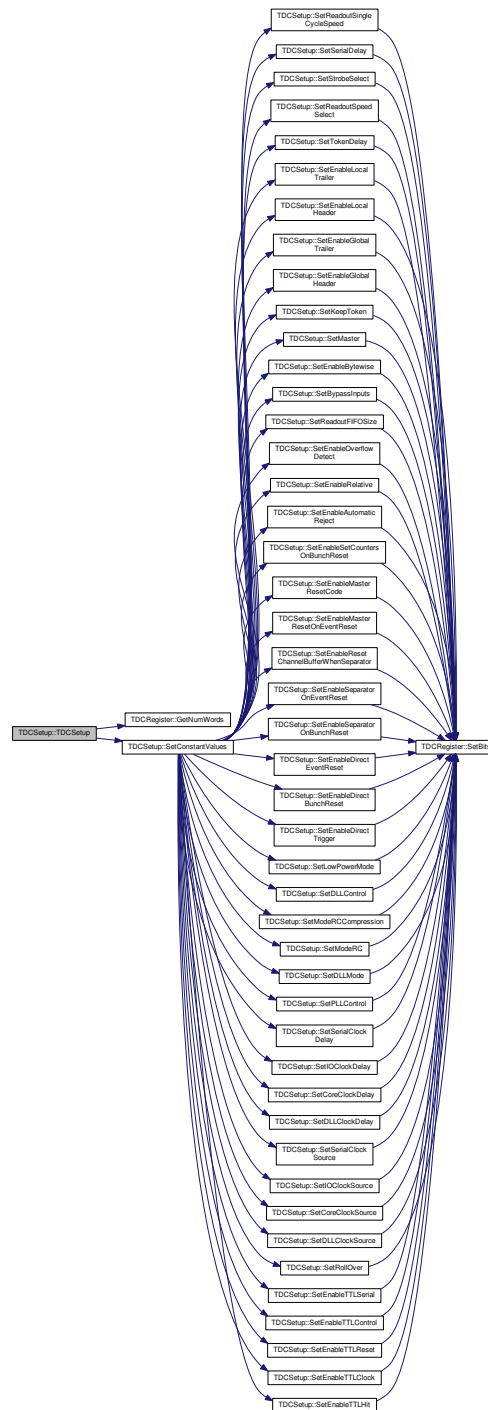
5.15.2.1 TDCSetup::TDCSetup ()

Here is the call graph for this function:



5.15.2.2 TDCSetup::TDCSetup (const TDCSetup & c)

Here is the call graph for this function:

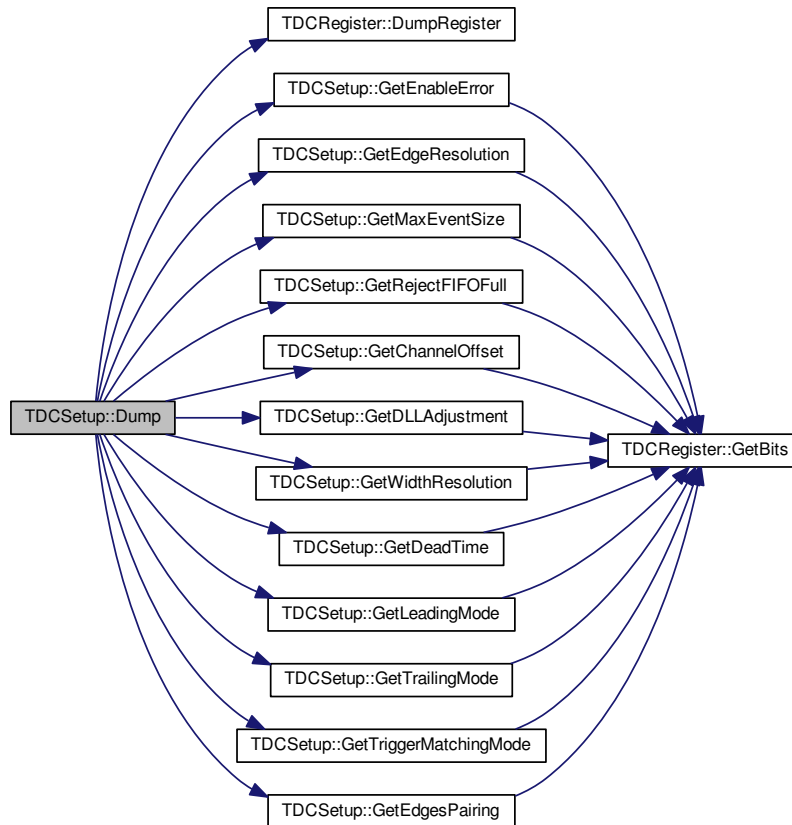


5.15.2.3 TDCSetup::~TDCSetup () [virtual]

5.15.3 Member Function Documentation

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

Here is the call graph for this function:



5.15.3.2 `uint16_t TDCSetup::GetChannelOffset (int channel) const` `[inline]`

Return the offset for one single channel.

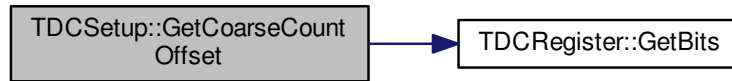
Here is the call graph for this function:



5.15.3.3 `uint16_t TDCSetup::GetCoarseCountOffset () const` `[inline]`

Extract offset for the coarse time counter.

Here is the call graph for this function:



5.15.3.4 DeadTime TDCSetup::GetDeadTime () const [inline]

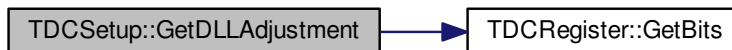
Here is the call graph for this function:



5.15.3.5 uint8_t TDCSetup::GetDLLAdjustment (int tap) const [inline]

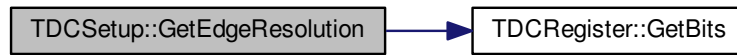
Set the adjustment of DLL taps.

Here is the call graph for this function:



5.15.3.6 **EdgeResolution** TDCSetup::GetEdgeResolution () const [inline]

Here is the call graph for this function:



5.15.3.7 **bool** TDCSetup::GetEdgesPairing () const [inline]

Here is the call graph for this function:



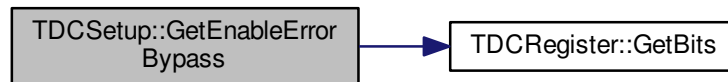
5.15.3.8 **uint16_t** TDCSetup::GetEnableError () const [inline]

Here is the call graph for this function:



5.15.3.9 bool TDCSetup::GetEnableErrorBypass () const [inline]

Here is the call graph for this function:



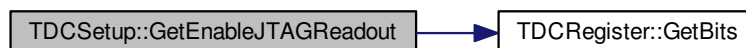
5.15.3.10 bool TDCSetup::GetEnableErrorMark () const [inline]

Here is the call graph for this function:



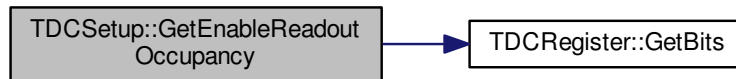
5.15.3.11 bool TDCSetup::GetEnableJTAGReadout () const [inline]

Here is the call graph for this function:



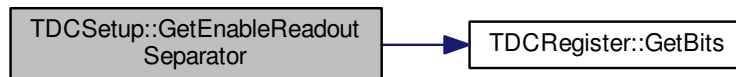
5.15.3.12 `bool TDCSetup::GetEnableReadoutOccupancy () const [inline]`

Here is the call graph for this function:



5.15.3.13 `bool TDCSetup::GetEnableReadoutSeparator () const [inline]`

Here is the call graph for this function:



5.15.3.14 `bool TDCSetup::GetEnableSerial () const [inline]`

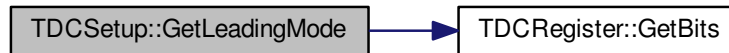
Here is the call graph for this function:



5.15.3.15 `bool TDCSetup::GetLeadingMode () const [inline]`

Extract the status for the detection of leading edges.

Here is the call graph for this function:



5.15.3.16 `uint16_t TDCSetup::GetMatchWindow () const [inline]`

Extract the matching window (in multiples of clock cycles: 0=25 ns, 1=50 ns, ...)

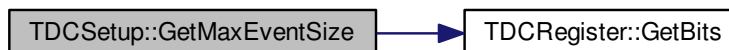
Here is the call graph for this function:



5.15.3.17 `uint8_t TDCSetup::GetMaxEventSize () const [inline]`

Extract the maximum number of hits per event.

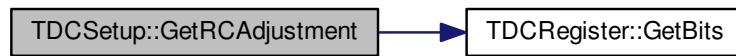
Here is the call graph for this function:



5.15.3.18 `uint8_t TDCSetup::GetRCAdjustment (int tap) [inline]`

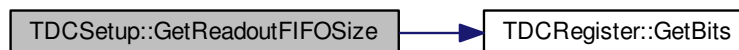
Extract the adjustment of the RC delay line.

Here is the call graph for this function:



5.15.3.19 `int TDCSetup::GetReadoutFIFOSize () const [inline]`

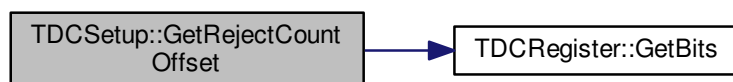
Here is the call graph for this function:



5.15.3.20 `uint16_t TDCSetup::GetRejectCountOffset () const [inline]`

Extract the offset in reject counter.

Here is the call graph for this function:



5.15.3.21 `bool TDCSetup::GetRejectFIFOFull () const [inline]`

Are hits rejected when readout FIFO is full?

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

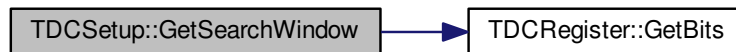
Here is the call graph for this function:



5.15.3.22 `uint16_t TDCSetup::GetSearchWindow () const [inline]`

Extract the search window (in multiples of clock cycles: 0=25 ns, 1=50 ns, ...)

Here is the call graph for this function:



5.15.3.23 `bool TDCSetup::GetSetupParity () const [inline]`

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

Here is the call graph for this function:



5.15.3.24 `bool TDCSetup::GetTestInvert () const [inline]`

Here is the call graph for this function:



5.15.3.25 `bool TDCSetup::GetTestMode () const [inline]`

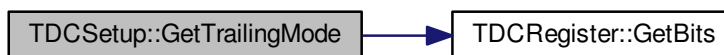
Here is the call graph for this function:



5.15.3.26 `bool TDCSetup::GetTrailingMode () const [inline]`

Extract the status for the detection of trailing edges.

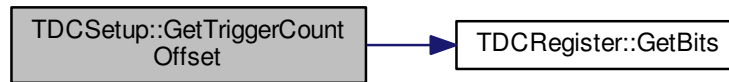
Here is the call graph for this function:



5.15.3.27 `uint16_t TDCSetup::GetTriggerCountOffset () const [inline]`

Extract trigger time tag count offset.

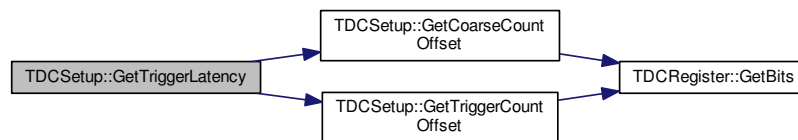
Here is the call graph for this function:



5.15.3.28 `uint16_t TDCSetup::GetTriggerLatency () const [inline]`

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

Here is the call graph for this function:



5.15.3.29 `bool TDCSetup::GetTriggerMatchingMode () const [inline]`

Extract the enable status of trigger matching mode.

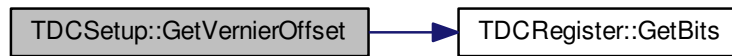
Here is the call graph for this function:



5.15.3.30 `uint8_t TDCSetup::GetVernierOffset () const [inline]`

Extract the offset in vernier decoding.

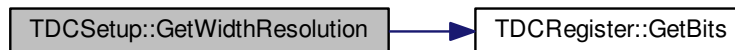
Here is the call graph for this function:



5.15.3.31 WidthResolution TDCSetup::GetWidthResolution () const [inline]

Extract the pulse width resolution when paired measurements are performed.

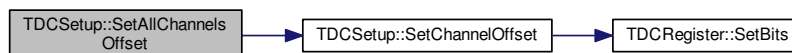
Here is the call graph for this function:



5.15.3.32 void TDCSetup::SetAllChannelsOffset (uint16_t offset) [inline]

Set the time offset for all channels.

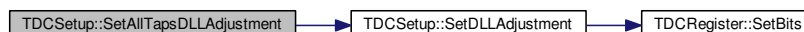
Here is the call graph for this function:



5.15.3.33 void TDCSetup::SetAllTapsDLLAdjustment (uint8_t adj) [inline]

Extract the adjustment of DLL taps.

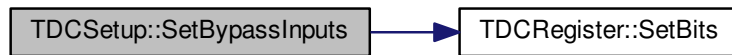
Here is the call graph for this function:



5.15.3.34 `void TDCSetup::SetBypassInputs (const bool sbi = true) [inline], [private]`

Select serial in and token in from bypass inputs.

Here is the call graph for this function:



5.15.3.35 `void TDCSetup::SetChannelOffset (int channel, uint16_t offset) [inline]`

Set the time offset for one single channel.

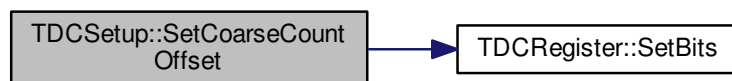
Here is the call graph for this function:



5.15.3.36 `void TDCSetup::SetCoarseCountOffset (uint16_t cco) [inline]`

Set offset for the coarse time counter.

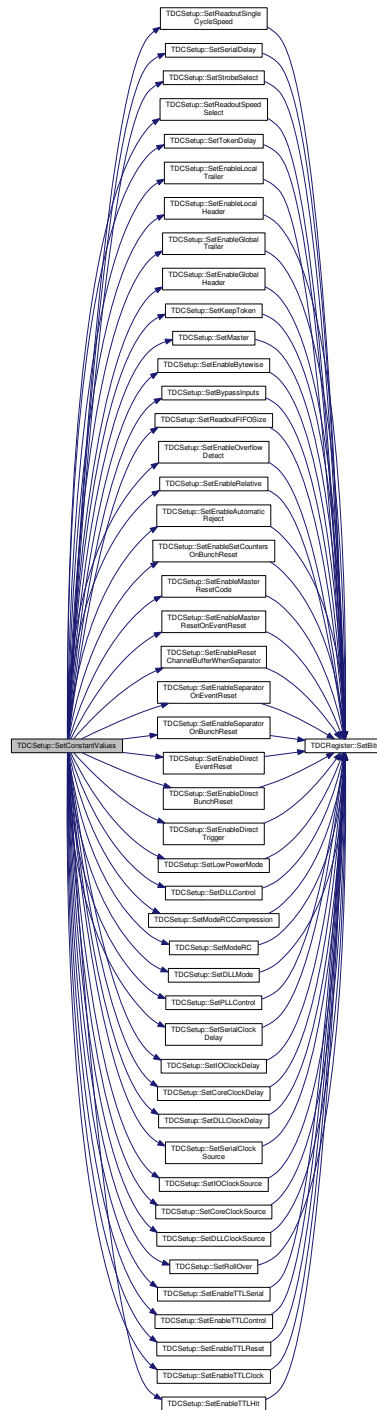
Here is the call graph for this function:



5.15.3.37 `void TDCSetup::SetConstantValues ()`

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

Here is the call graph for this function:



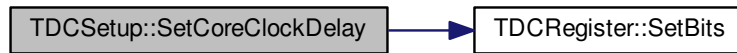
5.15.3.38 `void TDCSetup::SetCoreClockDelay (const bool delay_clock, const uint8_t delay)` `[inline], [private]`

Delay of internal core clock.

Parameters

in	<i>delay_clock</i>	Use of direct clock (0) or delayed clock (1)
in	<i>delay</i>	Delay in steps of (typically) 0.13 ns

Here is the call graph for this function:



5.15.3.39 `void TDCSetup::SetCoreClockSource (const CoreClockSource ccs)` `[inline]`, `[private]`

Selection of clock source for internal logic.

Here is the call graph for this function:



5.15.3.40 `void TDCSetup::SetDeadTime (const DeadTime dt)` `[inline]`

Channel dead time between hits.

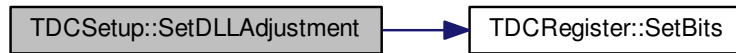
Here is the call graph for this function:



5.15.3.41 `void TDCSetup::SetDLLAdjustment (int tap, uint8_t adj)` `[inline]`

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

Here is the call graph for this function:



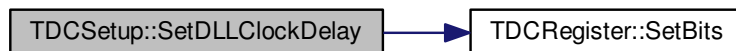
5.15.3.42 void TDCSetup::SetDLLClockDelay (const bool *delay_clock*, const uint8_t *delay*) [inline],[private]

Delay of internal DLL clock.

Parameters

in	<i>delay_clock</i>	Use of direct clock (0) or delayed clock (1)
in	<i>delay</i>	Delay in steps of (typically) 0.13 ns

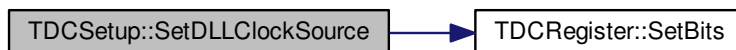
Here is the call graph for this function:



5.15.3.43 void TDCSetup::SetDLLClockSource (const DLLClockSource *dcs*) [inline],[private]

Selection of clock source for DLL.

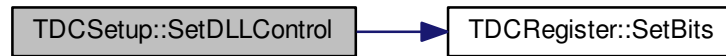
Here is the call graph for this function:



5.15.3.44 void TDCSetup::SetDLLControl (const uint8_t *dc*) [inline],[private]

Control of DLL (DLL charge pump levels)

Here is the call graph for this function:



5.15.3.45 `void TDCSetup::SetDLLMode (const DLLSpeedMode dsm)` `[inline]`, `[private]`

Selection of DLL speed mode.

Here is the call graph for this function:



5.15.3.46 `void TDCSetup::SetEdgeResolution (const EdgeResolution r)` `[inline]`

Here is the call graph for this function:



5.15.3.47 `void TDCSetup::SetEdgesPairing (const bool pair = true)` `[inline]`

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

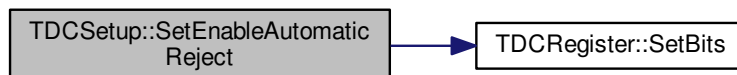
Here is the call graph for this function:



5.15.3.48 `void TDCSetup::SetEnableAutomaticReject (const bool ear = true) [inline],[private]`

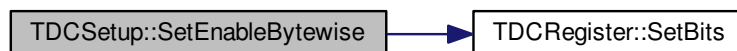
Enable of automatic rejection (should always be enabled if trigger matching mode!)

Here is the call graph for this function:



5.15.3.49 `void TDCSetup::SetEnableBytewise (const bool seb = true) [inline],[private]`

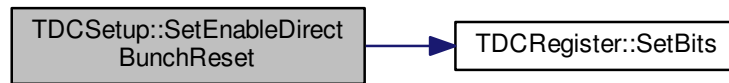
Here is the call graph for this function:



5.15.3.50 `void TDCSetup::SetEnableDirectBunchReset (const bool edbr = true) [inline],[private]`

Enable of direct bunch reset input pin (1), otherwise taken from encoded control.

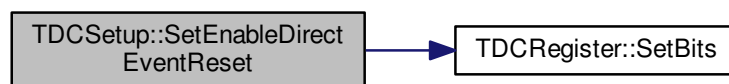
Here is the call graph for this function:



5.15.3.51 `void TDCSetup::SetEnableDirectEventReset (const bool eder = true) [inline], [private]`

Enable of direct event reset input pin (1), otherwise taken from encoded control.

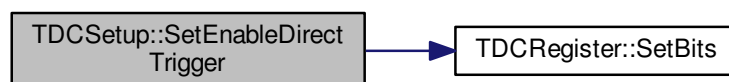
Here is the call graph for this function:



5.15.3.52 `void TDCSetup::SetEnableDirectTrigger (const bool edt = true) [inline], [private]`

Enable of direct trigger input pin.

Here is the call graph for this function:



5.15.3.53 `void TDCSetup::SetEnableError (const uint16_t & err) [inline]`

Enable internal error types for generation of global error signals.

Here is the call graph for this function:



5.15.3.54 `void TDCSetup::SetEnableErrorBypass (const bool eb) [inline]`

Bypass TDC chip if global error signal is set.

Here is the call graph for this function:



5.15.3.55 `void TDCSetup::SetEnableErrorMark (const bool em) [inline]`

Mark events with error if global error signal is set.

Here is the call graph for this function:



5.15.3.56 `void TDCSetup::SetEnableGlobalHeader (const bool egh = true) [inline], [private]`

Enable of global headers in read-out (only valid for master TDC)

Here is the call graph for this function:



5.15.3.57 `void TDCSetup::SetEnableGlobalTrailer (const bool egt = true) [inline],[private]`

Enable of global trailers in read-out (only valid for master TDC)

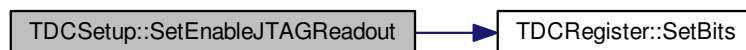
Here is the call graph for this function:



5.15.3.58 `void TDCSetup::SetEnableJTAGReadout (const bool jr) [inline]`

Enable of read-out via JTAG.

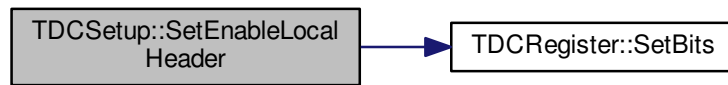
Here is the call graph for this function:



5.15.3.59 `void TDCSetup::SetEnableLocalHeader (const bool elh = true) [inline],[private]`

Enable of local headers in read-out.

Here is the call graph for this function:



5.15.3.60 `void TDCSetup::SetEnableLocalTrailer (const bool elt = true) [inline],[private]`

Enable of local trailers in read-out.

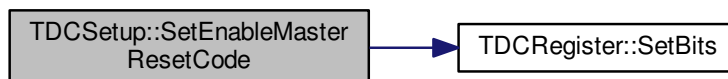
Here is the call graph for this function:



5.15.3.61 `void TDCSetup::SetEnableMasterResetCode (const bool emrc = true) [inline],[private]`

Enable master reset code on encoded_control.

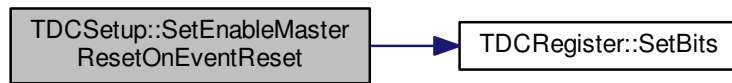
Here is the call graph for this function:



5.15.3.62 `void TDCSetup::SetEnableMasterResetOnEventReset (const bool emroer = true) [inline],[private]`

Enable master reset of whole TDC on event reset.

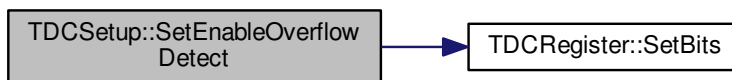
Here is the call graph for this function:



5.15.3.63 `void TDCSetup::SetEnableOverflowDetect (const bool eod = true) [inline], [private]`

Enable overflow detection of L1 buffers (should always be enabled!)

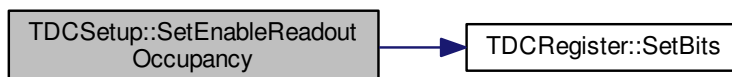
Here is the call graph for this function:



5.15.3.64 `void TDCSetup::SetEnableReadoutOccupancy (const bool ro = true) [inline]`

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

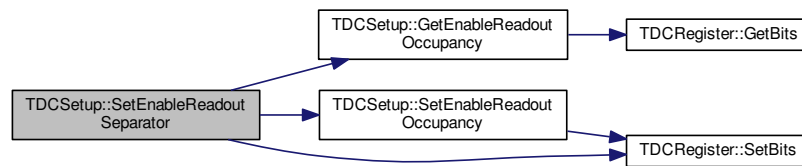
Here is the call graph for this function:



5.15.3.65 `void TDCSetup::SetEnableReadoutSeparator (const bool ro = true) [inline]`

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

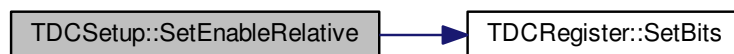
Here is the call graph for this function:



5.15.3.66 `void TDCSetup::SetEnableRelative (const bool er = true) [inline], [private]`

Enable read-out of relative time to trigger time tag. Only valid when using trigger matching mode.

Here is the call graph for this function:



5.15.3.67 `void TDCSetup::SetEnableResetChannelBufferWhenSeparator (const bool ercbws = true) [inline], [private]`

Enable reset channel buffers when separator.

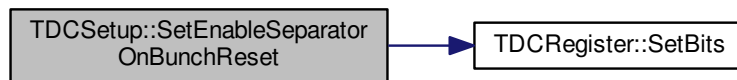
Here is the call graph for this function:



5.15.3.68 `void TDCSetup::SetEnableSeparatorOnBunchReset (const bool esobr = true) [inline], [private]`

Enable generation of separator on bunch reset.

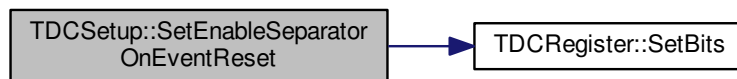
Here is the call graph for this function:



5.15.3.69 `void TDCSetup::SetEnableSeparatorOnEventReset (const bool esoe = true) [inline], [private]`

Enable generation of separator on event reset.

Here is the call graph for this function:



5.15.3.70 `void TDCSetup::SetEnableSerial (const bool es) [inline]`

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

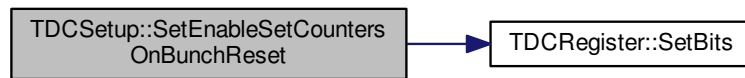
Here is the call graph for this function:



5.15.3.71 `void TDCSetup::SetEnableSetCountersOnBunchReset (const bool escobr = true) [inline], [private]`

Enable all counters to be set on bunch count reset.

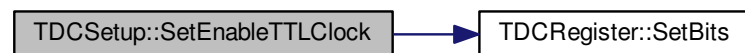
Here is the call graph for this function:



5.15.3.72 `void TDCSetup::SetEnableTTLClock (const bool tc = true) [inline], [private]`

Enable LV TTL inputs on: clk, aux_clock, otherwise uses LVDS input levels.

Here is the call graph for this function:



5.15.3.73 `void TDCSetup::SetEnableTTLControl (const bool tc = true) [inline], [private]`

Enable LV TTL inputs on control registers.

Enable LV TTL input on:

- trigger,
- bunch_reset,
- event_reset,
- encoded_control, otherwise uses LVDS input levels.

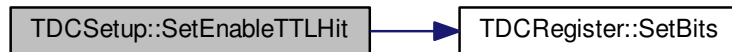
Here is the call graph for this function:



5.15.3.74 `void TDCSetup::SetEnableTTLHit (const bool th = true) [inline], [private]`

Enable LV TTL input on hit[31:0], otherwise uses LVDS input levels.

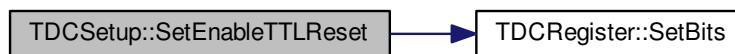
Here is the call graph for this function:



5.15.3.75 `void TDCSetup::SetEnableTTLReset (const bool tr = true) [inline], [private]`

Enable LV TTL input on reset, otherwise uses LVDS input levels.

Here is the call graph for this function:



5.15.3.76 `void TDCSetup::SetEnableTTLSerial (const bool ts = true) [inline], [private]`

Enable LV TTL inputs on serial registers, and disable their drivers.

Enable LV TTL input on:

- serial_in,
- serial_bypass_in,
- token_in,
- token_bypass_in, otherwise uses LVDS input levels. Disable LVDS drivers on:
- serial_out,
- strobe_out,
- token_out.

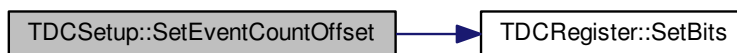
Here is the call graph for this function:



5.15.3.77 void TDCSetup::SetEventCountOffset (uint16_t *eco*) [inline]

Set offset for the event counter.

Here is the call graph for this function:



5.15.3.78 void TDCSetup::SetIOClockDelay (const bool *delay_clock*, const uint8_t *delay*) [inline],[private]

Delay of internal I/O clock.

Parameters

in	<i>delay_clock</i>	Use of direct clock (0) or delayed clock (1)
in	<i>delay</i>	Delay in steps of (typically) 0.13 ns

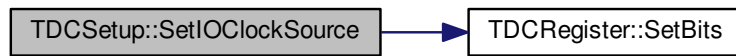
Here is the call graph for this function:



5.15.3.79 void TDCSetup::SetIOClockSource (const IOClockSource *ics*) [inline],[private]

Selection of clock source for I/O signals.

Here is the call graph for this function:



5.15.3.80 `void TDCSetup::SetKeepToken (const bool kt = true) [inline], [private]`

Keep token until end of event or no more data, otherwise pass token after each word read. Must be enabled when using trigger matching.

Here is the call graph for this function:



5.15.3.81 `void TDCSetup::SetLeadingMode (const bool lead = true) [inline]`

Enable the detection of leading edges.

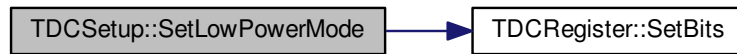
Here is the call graph for this function:



5.15.3.82 `void TDCSetup::SetLowPowerMode (const bool lpm = true) [inline], [private]`

Low power mode of channel buffers.

Here is the call graph for this function:



5.15.3.83 `void TDCSetup::SetMaster (const bool m=true) [inline],[private]`

Here is the call graph for this function:



5.15.3.84 `void TDCSetup::SetMatchWindow (uint16_t mw) [inline]`

Set the matching window (in multiples of clock cycles: 0=25 ns, 1=50 ns, ...)

Here is the call graph for this function:

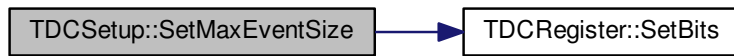


5.15.3.85 `void TDCSetup::SetMaxEventSize (int sz=-1) [inline]`

Set the maximum number of hits per event.

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

Here is the call graph for this function:



5.15.3.86 `void TDCSetup::SetModeRC (const bool mr = true) [inline], [private]`

Enable of RR delay lines mode (in very high resolution mode) ; only for channels 0-4-8-12-16-20-24-28 active.

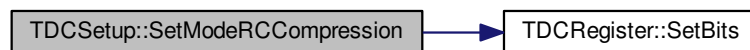
Here is the call graph for this function:



5.15.3.87 `void TDCSetup::SetModeRCCompression (const bool mrc = true) [inline], [private]`

Perform RC interpolation on-chip (only valid in very high resolution mode)

Here is the call graph for this function:



5.15.3.88 `void TDCSetup::SetPLLControl (const uint8_t charge_pump_current = 0x4, const bool power_down_mode = false, const bool enable_test_outputs = false, const bool invert_connection_to_status = false) [inline], [private]`

Control of PLL.

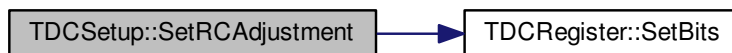
Here is the call graph for this function:



5.15.3.89 `void TDCSetup::SetRCAdjustment (int tap, uint8_t adj)` `[inline]`

Set the adjustment of the RC delay line.

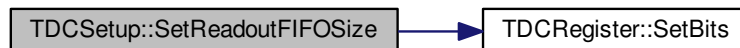
Here is the call graph for this function:



5.15.3.90 `void TDCSetup::SetReadoutFIFOSize (int rfs)` `[inline]`

Effective size of readout FIFO.

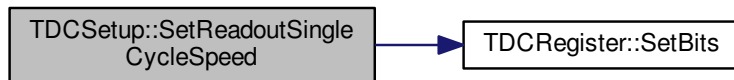
Here is the call graph for this function:



5.15.3.91 `void TDCSetup::SetReadoutSingleCycleSpeed (const ReadoutSingleCycleSpeed rscs = RSC_40Mbits_s)`
`[inline], [private]`

Serial transmission speed in single cycle mode.

Here is the call graph for this function:



5.15.3.92 `void TDCSetup::SetReadoutSpeedSelect (const ReadoutSpeed rss = RO_Fixed) [inline], [private]`

Selection of serial read-out speed.

Parameters

in	rss	
		<ul style="list-style-type: none"> • 0: Selection of serial read-out speed (as defined by setup[19:17], <i>SetReadoutSingleCycleSpeed</i>) • 1: 80 Mbits/s (PLL lock required)

Here is the call graph for this function:



5.15.3.93 `void TDCSetup::SetRejectCountOffset (uint16_t rco) [inline]`

Set the offset in reject counter (defines reject latency together with coarse count offset)

Here is the call graph for this function:

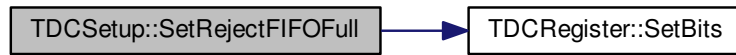


5.15.3.94 `void TDCSetup::SetRejectFIFOFull (const bool rej = true) [inline]`

Reject hits when readout FIFO full.

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

Here is the call graph for this function:



5.15.3.95 `void TDCSetup::SetRollOver (const uint16_t ro = 0xFFFF) [inline],[private]`

Counter roll over value, defining maximal count value from where counters will be reset to 0.

Here is the call graph for this function:



5.15.3.96 `void TDCSetup::SetSearchWindow (uint16_t sw) [inline]`

Set the search window (in multiples of clock cycles: 0=25 ns, 1=50 ns, ...)

Here is the call graph for this function:



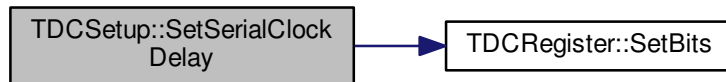
5.15.3.97 `void TDCSetup::SetSerialClockDelay (const bool delay_clock, const uint8_t delay) [inline],[private]`

Delay of internal serial clock.

Parameters

in	<i>delay_clock</i>	Use of direct clock (0) or delayed clock (1)
in	<i>delay</i>	Delay in steps of (typically) 0.13 ns

Here is the call graph for this function:



5.15.3.98 `void TDCSetup::SetSerialClockSource (const SerialClockSource scs) [inline], [private]`

Selection of source for serial clock.

Here is the call graph for this function:



5.15.3.99 `void TDCSetup::SetSerialDelay (const uint8_t sd = 0x0) [inline], [private]`

Programmable delay of serial input, in time unit ~ 1 ns.

Here is the call graph for this function:



5.15.3.100 `void TDCSetup::SetSetupParity (const bool sp = true) [inline]`

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

Here is the call graph for this function:



5.15.3.101 `void TDCSetup::SetStrobeSelect (const SerialStrobeType ss = SS_NoStrobe) [inline], [private]`

Here is the call graph for this function:



5.15.3.102 `void TDCSetup::SetTestInvert (const bool ti=true) [inline]`

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

Here is the call graph for this function:



5.15.3.103 `void TDCSetup::SetTestMode (const bool tm=true) [inline]`

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

Here is the call graph for this function:



5.15.3.104 `void TDCSetup::SetTokenDelay (const uint8_t td = 0x0) [inline], [private]`

Programmable delay of token input, in time unit ~ 1 ns.

Here is the call graph for this function:



5.15.3.105 `void TDCSetup::SetTrailingMode (const bool trail = true) [inline]`

Enable/disable the detection of trailing edges.

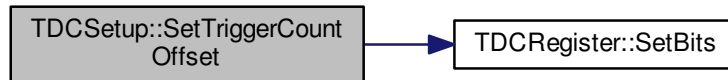
Here is the call graph for this function:



5.15.3.106 `void TDCSetup::SetTriggerCountOffset (uint16_t tco) [inline]`

Set offset for the trigger time tag counter to set effective trigger latency.

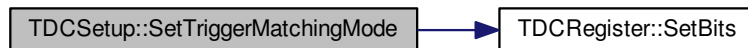
Here is the call graph for this function:



5.15.3.107 `void TDCSetup::SetTriggerMatchingMode (const bool trig = true) [inline]`

Set the enable status of trigger matching mode.

Here is the call graph for this function:



5.15.3.108 `void TDCSetup::SetVernierOffset (const uint8_t vo) [inline]`

Set the offset in vernier decoding.

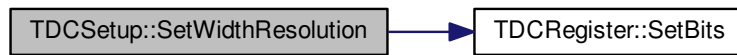
Here is the call graph for this function:



5.15.3.109 `void TDCSetup::SetWidthResolution (const WidthResolution r) [inline]`

Set the pulse width resolution when paired measurements are performed.

Here is the call graph for this function:



5.15.4 Field Documentation

- 5.15.4.1 `const bit TDCSetup::kCoarseCountOffset = 447` [static], [private]
- 5.15.4.2 `const bit TDCSetup::kCoreClockDelay = 610` [static], [private]
- 5.15.4.3 `const bit TDCSetup::kCoreClockSource = 622` [static], [private]
- 5.15.4.4 `const bit TDCSetup::kDeadTime = 584` [static], [private]
- 5.15.4.5 `const bit TDCSetup::kDLLClockDelay = 614` [static], [private]
- 5.15.4.6 `const bit TDCSetup::kDLLClockSource = 624` [static], [private]
- 5.15.4.7 `const bit TDCSetup::kDLLControl = 580` [static], [private]
- 5.15.4.8 `const bit TDCSetup::kDLLMode = 592` [static], [private]
- 5.15.4.9 `const bit TDCSetup::kDLLTapAdjust0 = 459` [static], [private]
- 5.15.4.10 `const bit TDCSetup::kEnableAutomaticReject = 125` [static], [private]
- 5.15.4.11 `const bit TDCSetup::kEnableBytewise = 37` [static], [private]
- 5.15.4.12 `const bit TDCSetup::kEnableDirectBunchReset = 157` [static], [private]
- 5.15.4.13 `const bit TDCSetup::kEnableDirectEventReset = 156` [static], [private]
- 5.15.4.14 `const bit TDCSetup::kEnableDirectTrigger = 158` [static], [private]
- 5.15.4.15 `const bit TDCSetup::kEnableError = 6` [static], [private]
- 5.15.4.16 `const bit TDCSetup::kEnableErrorBypass = 5` [static], [private]
- 5.15.4.17 `const bit TDCSetup::kEnableErrorMark = 4` [static], [private]
- 5.15.4.18 `const bit TDCSetup::kEnableGlobalHeader = 34` [static], [private]
- 5.15.4.19 `const bit TDCSetup::kEnableGlobalTrailer = 33` [static], [private]
- 5.15.4.20 `const bit TDCSetup::kEnableJTAGReadout = 39` [static], [private]
- 5.15.4.21 `const bit TDCSetup::kEnableLocalHeader = 32` [static], [private]

5.15.4.22 `const bit TDCSetup::kEnableLocalTrailer = 31` [static], [private]

5.15.4.23 `const bit TDCSetup::kEnableMasterResetCode = 151` [static], [private]

5.15.4.24 `const bit TDCSetup::kEnableMasterResetOnEventReset = 152` [static], [private]

5.15.4.25 `const bit TDCSetup::kEnableMatching = 639` [static], [private]

5.15.4.26 `const bit TDCSetup::kEnableOverflowDetect = 123` [static], [private]

5.15.4.27 `const bit TDCSetup::kEnablePair = 640` [static], [private]

5.15.4.28 `const bit TDCSetup::kEnableReadoutOccupancy = 121` [static], [private]

5.15.4.29 `const bit TDCSetup::kEnableReadoutSeparator = 122` [static], [private]

5.15.4.30 `const bit TDCSetup::kEnableRelative = 124` [static], [private]

5.15.4.31 `const bit TDCSetup::kEnableResetChannelBufferWhenSeparator = 153` [static], [private]

5.15.4.32 `const bit TDCSetup::kEnableSeparatorOnBunchReset = 155` [static], [private]

5.15.4.33 `const bit TDCSetup::kEnableSeparatorOnEventReset = 154` [static], [private]

5.15.4.34 `const bit TDCSetup::kEnableSerial = 38` [static], [private]

5.15.4.35 `const bit TDCSetup::kEnableSetCountersOnBunchReset = 150` [static], [private]

5.15.4.36 `const bit TDCSetup::kEnableTTLClock = 644` [static], [private]

5.15.4.37 `const bit TDCSetup::kEnableTTLControl = 642` [static], [private]

5.15.4.38 `const bit TDCSetup::kEnableTTLHit = 645` [static], [private]

5.15.4.39 `const bit TDCSetup::kEnableTTLReset = 643` [static], [private]

5.15.4.40 `const bit TDCSetup::kEnableTTLSerial = 641` [static], [private]

5.15.4.41 `const bit TDCSetup::kEventCountOffset = 126` [static], [private]

5.15.4.42 `const bit TDCSetup::kIOClockDelay = 606` [static], [private]

5.15.4.43 `const bit TDCSetup::kIOClockSource = 620` [static], [private]

5.15.4.44 `const bit TDCSetup::kKeepToken = 35` [static], [private]

5.15.4.45 `const bit TDCSetup::kLeading = 589` [static], [private]

5.15.4.46 `const bit TDCSetup::kLeadingResolution = 84` [static], [private]

5.15.4.47 `const bit TDCSetup::kLowPowerMode = 570` [static], [private]

5.15.4.48 `const bit TDCSetup::kMaster = 36` [static], [private]

5.15.4.49 `const bit TDCSetup::kMatchWindow = 72` [static], [private]

- 5.15.4.50 `const bit TDCSetup::kMaxEventSize = 116` [static], [private]
- 5.15.4.51 `const bit TDCSetup::kModeRC = 591` [static], [private]
- 5.15.4.52 `const bit TDCSetup::kModeRCCompression = 590` [static], [private]
- 5.15.4.53 `const bit TDCSetup::kOffset0 = 438` [static], [private]
- 5.15.4.54 `const bit TDCSetup::kPLLControl = 594` [static], [private]
- 5.15.4.55 `const bit TDCSetup::kRCAdjust0 = 555` [static], [private]
- 5.15.4.56 `const bit TDCSetup::kReadoutFIFOSize = 45` [static], [private]
- 5.15.4.57 `const bit TDCSetup::kReadoutSingleCycleSpeed = 17` [static], [private]
- 5.15.4.58 `const bit TDCSetup::kReadoutSpeedSelect = 26` [static], [private]
- 5.15.4.59 `const bit TDCSetup::kRejectCountOffset = 48` [static], [private]
- 5.15.4.60 `const bit TDCSetup::kRejectFIFOFull = 120` [static], [private]
- 5.15.4.61 `const bit TDCSetup::kRollOver = 627` [static], [private]
- 5.15.4.62 `const bit TDCSetup::kSearchWindow = 60` [static], [private]
- 5.15.4.63 `const bit TDCSetup::kSelectBypassInputs = 44` [static], [private]
- 5.15.4.64 `const bit TDCSetup::kSerialClockDelay = 602` [static], [private]
- 5.15.4.65 `const bit TDCSetup::kSerialClockSource = 618` [static], [private]
- 5.15.4.66 `const bit TDCSetup::kSerialDelay = 20` [static], [private]
- 5.15.4.67 `const bit TDCSetup::kSetupParity = 646` [static], [private]
- 5.15.4.68 `const bit TDCSetup::kStrobeSelect = 24` [static], [private]
- 5.15.4.69 `const bit TDCSetup::kTDCId = 40` [static], [private]
- 5.15.4.70 `const bit TDCSetup::kTestInvert = 586` [static], [private]
- 5.15.4.71 `const bit TDCSetup::kTestMode = 587` [static], [private]
- 5.15.4.72 `const bit TDCSetup::kTestSelect = 0` [static], [private]
- 5.15.4.73 `const bit TDCSetup::kTokenDelay = 27` [static], [private]
- 5.15.4.74 `const bit TDCSetup::kTrailing = 588` [static], [private]
- 5.15.4.75 `const bit TDCSetup::kTriggerCountOffset = 138` [static], [private]
- 5.15.4.76 `const bit TDCSetup::kVernierOffset = 575` [static], [private]

5.15.4.77 `const bit TDCSetup::kWidthSelect = 571` `[static], [private]`

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

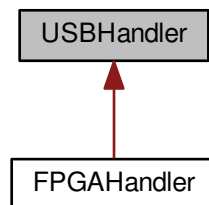
- `include/TDCSetup.h`
- `src/TDCSetup.cpp`

5.16 USBHandler Class Reference

Generic USB communication handler.

```
#include <USBHandler.h>
```

Inheritance diagram for USBHandler:



Public Member Functions

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

Protected Member Functions

- void `Write` (`uint32_t word`, `uint8_t size`) `const`
Write a word to the USB device.
- `uint32_t Fetch` (`uint8_t size`) `const`
Receive a word from the USB device.

Private Attributes

- `std::string fDevice`
- `libusb_device_handle * fHandle`

5.16.1 Detailed Description

Generic USB communication handler.

Date

21 Apr 2015

Author

Laurent Forthomme laurent.forthomme@cern.ch

5.16.2 Constructor & Destructor Documentation

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

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

5.16.3 Member Function Documentation

5.16.3.1 void USBHandler::DumpDevice (libusb_device * *dev*, int *verb* = 1, std::ostream & *out* = std::cout)5.16.3.2 uint32_t USBHandler::Fetch (uint8_t *size*) const [inline],[protected]

Receive a word from the USB device.

5.16.3.3 void USBHandler::Init ()

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

A libusb session

Here is the call graph for this function:

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

Write a word to the USB device.

5.16.4 Field Documentation

5.16.4.1 std::string USBHandler::fDevice [private]

5.16.4.2 libusb_device_handle* USBHandler::fHandle [private]

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

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

Index

- ~Client
 - Client, [15](#)
- ~Exception
 - Exception, [19](#)
- ~FPGAHandler
 - FPGAHandler, [24](#)
- ~Message
 - Message, [31](#)
- ~Messenger
 - Messenger, [33](#)
- ~Socket
 - Socket, [41](#)
- ~SocketMessage
 - SocketMessage, [49](#)
- ~TDCBoundaryScanRegister
 - TDCBoundaryScanRegister, [52](#)
- ~TDCControl
 - TDCControl, [54](#)
- ~TDCEvent
 - TDCEvent, [57](#)
- ~TDCRegister
 - TDCRegister, [61](#)
- ~TDCSetup
 - TDCSetup, [71](#)
- ~USBHandler
 - USBHandler, [113](#)
- AcceptConnections
 - Socket, [41](#)
- AddClient
 - Messenger, [34](#)
- Announce
 - Client, [15](#)
- Bind
 - Socket, [41](#)
- bit
 - TDCRegister, [61](#)
- Broadcast
 - Messenger, [34](#)
- CLIENT
 - Socket communication objects, [7](#)
- ChannelSelectError
 - HPTDC chip control, [11](#)
- Client, [13](#)
 - ~Client, [15](#)
 - Announce, [15](#)
 - Client, [15](#)
 - Connect, [16](#)
 - Disconnect, [16](#)
 - fClientId, [18](#)
 - flsConnected, [18](#)
 - GetType, [17](#)
 - ParseMessage, [17](#)
 - Receive, [17](#)
 - Send, [17](#)
- CloseFile
 - FPGAHandler, [24](#)
- CoarseError
 - HPTDC chip control, [11](#)
- config
 - file_header_t, [21](#)
- Configure
 - Socket, [41](#)
- Connect
 - Client, [16](#)
 - Messenger, [35](#)
- ControlParityError
 - HPTDC chip control, [11](#)
- Core_aux_clock
 - HPTDC chip control, [10](#)
- Core_clock_40
 - HPTDC chip control, [10](#)
- Core_pll_clock_160
 - HPTDC chip control, [10](#)
- Core_pll_clock_80
 - HPTDC chip control, [10](#)
- CoreClockSource
 - HPTDC chip control, [10](#)
- Create
 - Socket, [42](#)
- DETECTOR
 - Socket communication objects, [8](#)
- DLL_160MHz
 - HPTDC chip control, [10](#)
- DLL_320MHz
 - HPTDC chip control, [10](#)
- DLL_40MHz
 - HPTDC chip control, [10](#)
- DLL_Illegal
 - HPTDC chip control, [10](#)
- DLL_aux_clock
 - HPTDC chip control, [10](#)
- DLL_clock_40
 - HPTDC chip control, [10](#)
- DLL_pll_clock_160
 - HPTDC chip control, [10](#)
- DLL_pll_clock_320

- HPTDC chip control, [10](#)
- DLL_pll_clock_40
 - HPTDC chip control, [10](#)
- DLLClockSource
 - HPTDC chip control, [10](#)
- DLLSpeedMode
 - HPTDC chip control, [10](#)
- DT_100ns
 - HPTDC chip control, [10](#)
- DT_10ns
 - HPTDC chip control, [10](#)
- DT_30ns
 - HPTDC chip control, [10](#)
- DT_5ns
 - HPTDC chip control, [10](#)
- DeadTime
 - HPTDC chip control, [10](#)
- Debug
 - HPTDC chip control, [11](#)
- Decode
 - HTTPMessage, [29](#)
- Description
 - Exception, [19](#)
- Disconnect
 - Client, [16](#)
 - Messenger, [35](#)
- DisconnectClient
 - Messenger, [36](#)
- Dump
 - Exception, [19](#)
 - HTTPMessage, [29](#)
 - Message, [31](#)
 - SocketMessage, [49](#)
 - TDCControl, [54](#)
 - TDCSetup, [71](#)
- DumpConnected
 - Socket, [42](#)
- DumpDevice
 - USBHandler, [113](#)
- DumpRegister
 - TDCRegister, [61](#)
- E_100ps
 - HPTDC chip control, [10](#)
- E_12p5ns
 - HPTDC chip control, [10](#)
- E_1p6ns
 - HPTDC chip control, [10](#)
- E_200ps
 - HPTDC chip control, [10](#)
- E_3p12ns
 - HPTDC chip control, [10](#)
- E_400ps
 - HPTDC chip control, [10](#)
- E_6p25ns
 - HPTDC chip control, [10](#)
- E_800ps
 - HPTDC chip control, [10](#)
- EdgeResolution
 - HPTDC chip control, [10](#)
- EnablePattern
 - TDCControl, [54](#)
- EnabledError
 - HPTDC chip control, [10](#)
- Encode
 - HTTPMessage, [29](#)
- Error
 - HPTDC chip control, [11](#)
- ErrorNumber
 - Exception, [20](#)
- EventType
 - HPTDC chip control, [11](#)
- Exception, [18](#)
 - ~Exception, [19](#)
 - Description, [19](#)
 - Dump, [19](#)
 - ErrorNumber, [20](#)
 - Exception, [19](#)
 - fDescription, [20](#)
 - fErrorNumber, [20](#)
 - fFrom, [20](#)
 - fType, [20](#)
 - From, [20](#)
 - Type, [20](#)
 - TypeString, [20](#)
- fAddress
 - Socket, [44](#)
- fBuffer
 - Socket, [44](#)
- fClientId
 - Client, [18](#)
- fDescription
 - Exception, [20](#)
- fDevice
 - USBHandler, [113](#)
- fErrorNumber
 - Exception, [20](#)
- fFilename
 - FPGAHandler, [26](#)
- fFrom
 - Exception, [20](#)
- fHandle
 - USBHandler, [113](#)
- fIsConnected
 - Client, [18](#)
- fIsFileOpen
 - FPGAHandler, [26](#)
- fIsTDCInReadout
 - FPGAHandler, [26](#)
- fListenersInfo
 - Messenger, [38](#)
- fMaster
 - Socket, [44](#)
- fMessage
 - SocketMessage, [51](#)
- fNumAttempts
 - Messenger, [38](#)

- fNumWords
 - TDCRegister, 62
- fOriginalString
 - HTTPMessage, 29
- fOutput
 - FPGAHandler, 26
- FPGAHandler, 22
 - ~FPGAHandler, 24
 - CloseFile, 24
 - fFilename, 26
 - fIsFileOpen, 26
 - fIsTDCInReadout, 26
 - fOutput, 26
 - FPGAHandler, 24
 - fTDCBSR, 26
 - fTDCSetup, 26
 - GetConfiguration, 24
 - GetFilename, 24
 - GetType, 25
 - OpenFile, 25
 - ReadBoundaryScanRegister, 25
 - ReadBuffer, 25
 - ReadConfiguration, 25
 - SendConfiguration, 25
 - SetConfiguration, 26
- fPort
 - Socket, 44
- fReadFds
 - Socket, 44
- fSocketId
 - Socket, 44
- fSocketsConnected
 - Socket, 44
- fString
 - Message, 31
- fTDCBSR
 - FPGAHandler, 26
- fTDCSetup
 - FPGAHandler, 26
- fType
 - Exception, 20
- fWS
 - HTTPMessage, 29
 - Messenger, 38
- fWord
 - TDCEvent, 59
 - TDCRegister, 62
- Fetch
 - USBHandler, 113
- FetchMessage
 - Socket, 42
- file_header_t, 21
 - config, 21
 - magic, 21
 - run_id, 21
 - spill_id, 22
- From
 - Exception, 20
- GetBits
 - TDCRegister, 61
- GetBunchId
 - TDCEvent, 57
- GetChannelOffset
 - TDCSetup, 72
- GetCoarseCountOffset
 - TDCSetup, 72
- GetConfiguration
 - FPGAHandler, 24
- GetDLLAdjustment
 - TDCSetup, 73
- GetDeadTime
 - TDCSetup, 73
- GetEdgeResolution
 - TDCSetup, 73
- GetEdgesPairing
 - TDCSetup, 74
- GetEnableError
 - TDCSetup, 74
- GetEnableErrorBypass
 - TDCSetup, 74
- GetEnableErrorMark
 - TDCSetup, 75
- GetEnableJTAGReadout
 - TDCSetup, 75
- GetEnableReadoutOccupancy
 - TDCSetup, 75
- GetEnableReadoutSeparator
 - TDCSetup, 76
- GetEnableSerial
 - TDCSetup, 76
- GetErrorFlags
 - TDCEvent, 57
- GetEventId
 - TDCEvent, 58
- GetFilename
 - FPGAHandler, 24
- GetIntValue
 - SocketMessage, 49
- GetKey
 - HTTPMessage, 29
 - Message, 31
 - SocketMessage, 49
- GetLeadingMode
 - TDCSetup, 76
- GetLeadingTime
 - TDCEvent, 58
- GetMatchWindow
 - TDCSetup, 77
- GetMaxEventSize
 - TDCSetup, 77
- GetNumWords
 - TDCRegister, 61
- GetPort
 - Socket, 42
- GetRCAdjustment
 - TDCSetup, 77

- GetReadoutFIFOSize
 - TDCSetup, 78
- GetRejectCountOffset
 - TDCSetup, 78
- GetRejectFIFOFull
 - TDCSetup, 78
- GetSearchWindow
 - TDCSetup, 79
- GetSetupParity
 - TDCSetup, 79
- GetSocketId
 - Socket, 42
- GetSocketType
 - Socket, 42
- GetString
 - Message, 31
 - SocketMessage, 50
- GetTDCId
 - TDCEvent, 58
- GetTestInvert
 - TDCSetup, 79
- GetTestMode
 - TDCSetup, 80
- GetTrailingMode
 - TDCSetup, 80
- GetTrailingTime
 - TDCEvent, 58
- GetTriggerCountOffset
 - TDCSetup, 80
- GetTriggerLatency
 - TDCSetup, 81
- GetTriggerMatchingMode
 - TDCSetup, 81
- GetType
 - Client, 17
 - FPGAHandler, 25
 - Messenger, 36
 - TDCEvent, 59
- GetValue
 - SocketMessage, 50
- GetVectorValue
 - SocketMessage, 50
- GetVernierOffset
 - TDCSetup, 81
- GetWidth
 - TDCEvent, 59
- GetWidthResolution
 - TDCSetup, 82
- GetWord
 - TDCRegister, 61
- GetWordCount
 - TDCEvent, 59
- GroupHeader
 - HPTDC chip control, 11
- GroupTrailer
 - HPTDC chip control, 11
- HPTDC chip control, 9
 - ChannelSelectError, 11
- CoarseError, 11
- ControlParityError, 11
- Core_aux_clock, 10
- Core_clock_40, 10
- Core_pll_clock_160, 10
- Core_pll_clock_80, 10
- CoreClockSource, 10
- DLL_160MHz, 10
- DLL_320MHz, 10
- DLL_40MHz, 10
- DLL_Illegal, 10
- DLL_aux_clock, 10
- DLL_clock_40, 10
- DLL_pll_clock_160, 10
- DLL_pll_clock_320, 10
- DLL_pll_clock_40, 10
- DLLClockSource, 10
- DLLSpeedMode, 10
- DT_100ns, 10
- DT_10ns, 10
- DT_30ns, 10
- DT_5ns, 10
- DeadTime, 10
- Debug, 11
- E_100ps, 10
- E_12p5ns, 10
- E_1p6ns, 10
- E_200ps, 10
- E_3p12ns, 10
- E_400ps, 10
- E_6p25ns, 10
- E_800ps, 10
- EdgeResolution, 10
- EnabledError, 10
- Error, 11
- EventType, 11
- GroupHeader, 11
- GroupTrailer, 11
- IO_aux_clock, 11
- IO_clock_40, 11
- IO_pll_clock_160, 11
- IO_pll_clock_80, 11
- IOClockSource, 11
- Invalid, 11
- JTAGInstructionParityError, 11
- L1BufferParityError, 11
- LeadingEdge, 11
- RO_Fixed, 12
- RO_pll_80Mbits_s, 12
- RSC_10Mbits_s, 11
- RSC_1p25Mbits_s, 11
- RSC_20Mbits_s, 11
- RSC_2p5Mbits_s, 11
- RSC_312p5kbits_s, 11
- RSC_40Mbits_s, 11
- RSC_5Mbits_s, 11
- RSC_625kbits_s, 11
- ReadoutFIFOParityError, 11

- ReadoutSingleCycleSpeed, [11](#)
- ReadoutSpeed, [11](#)
- ReadoutStateError, [11](#)
- SS_DSStrobe, [12](#)
- SS_LeadingEdge, [12](#)
- SS_LeadingTrailingStrobe, [12](#)
- SS_NoStrobe, [12](#)
- Serial_aux_clock, [12](#)
- Serial_pll_clock_160, [12](#)
- Serial_pll_clock_40, [12](#)
- Serial_pll_clock_80, [12](#)
- SerialClockSource, [12](#)
- SerialStrobeType, [12](#)
- SetupParityError, [11](#)
- TDCHeader, [11](#)
- TDCTrailer, [11](#)
- TrailingEdge, [11](#)
- TriggerFIFOParityError, [11](#)
- TriggerMatchingError, [11](#)
- VernierError, [11](#)
- W_100ns, [12](#)
- W_100ps, [12](#)
- W_12p5ns, [12](#)
- W_1p6ns, [12](#)
- W_200ns, [12](#)
- W_200ps, [12](#)
- W_25ns, [12](#)
- W_3p2ns, [12](#)
- W_400ns, [12](#)
- W_400ps, [12](#)
- W_50ns, [12](#)
- W_6p25ns, [12](#)
- W_800ns, [12](#)
- W_800ps, [12](#)
- WidthResolution, [12](#)
- HTTPMessage, [27](#)
 - Decode, [29](#)
 - Dump, [29](#)
 - Encode, [29](#)
 - fOriginalString, [29](#)
 - fWS, [29](#)
 - GetKey, [29](#)
 - HTTPMessage, [28](#)
- INVALID
 - Socket communication objects, [7](#)
- IO_aux_clock
 - HPTDC chip control, [11](#)
- IO_clock_40
 - HPTDC chip control, [11](#)
- IO_pll_clock_160
 - HPTDC chip control, [11](#)
- IO_pll_clock_80
 - HPTDC chip control, [11](#)
- IOClockSource
 - HPTDC chip control, [11](#)
- Init
 - USBHandler, [113](#)
- Invalid
 - HPTDC chip control, [11](#)
 - IsFromWeb
 - Message, [31](#)
 - IsWebSocket
 - Socket, [42](#)
 - JTAGInstructionParityError
 - HPTDC chip control, [11](#)
 - kCoarseCountOffset
 - TDCSetup, [109](#)
 - kControlParity
 - TDCControl, [56](#)
 - kCoreClockDelay
 - TDCSetup, [109](#)
 - kCoreClockSource
 - TDCSetup, [109](#)
 - kDLLClockDelay
 - TDCSetup, [109](#)
 - kDLLClockSource
 - TDCSetup, [109](#)
 - kDLLControl
 - TDCSetup, [109](#)
 - kDLLMode
 - TDCSetup, [109](#)
 - kDLLReset
 - TDCControl, [56](#)
 - kDLLTapAdjust0
 - TDCSetup, [109](#)
 - kDeadTime
 - TDCSetup, [109](#)
 - kEnableAutomaticReject
 - TDCSetup, [109](#)
 - kEnableBytewise
 - TDCSetup, [109](#)
 - kEnableChannel
 - TDCControl, [56](#)
 - kEnableDirectBunchReset
 - TDCSetup, [109](#)
 - kEnableDirectEventReset
 - TDCSetup, [109](#)
 - kEnableDirectTrigger
 - TDCSetup, [109](#)
 - kEnableError
 - TDCSetup, [109](#)
 - kEnableErrorBypass
 - TDCSetup, [109](#)
 - kEnableErrorMark
 - TDCSetup, [109](#)
 - kEnableGlobalHeader
 - TDCSetup, [109](#)
 - kEnableGlobalTrailer
 - TDCSetup, [109](#)
 - kEnableJTAGReadout
 - TDCSetup, [109](#)
 - kEnableLocalHeader
 - TDCSetup, [109](#)
 - kEnableLocalTrailer
 - TDCSetup, [109](#)

- kEnableMasterResetCode
 - TDCSetup, [110](#)
- kEnableMasterResetOnEventReset
 - TDCSetup, [110](#)
- kEnableMatching
 - TDCSetup, [110](#)
- kEnableOverflowDetect
 - TDCSetup, [110](#)
- kEnablePair
 - TDCSetup, [110](#)
- kEnablePattern
 - TDCControl, [56](#)
- kEnableReadoutOccupancy
 - TDCSetup, [110](#)
- kEnableReadoutSeparator
 - TDCSetup, [110](#)
- kEnableRelative
 - TDCSetup, [110](#)
- kEnableResetChannelBufferWhenSeparator
 - TDCSetup, [110](#)
- kEnableSeparatorOnBunchReset
 - TDCSetup, [110](#)
- kEnableSeparatorOnEventReset
 - TDCSetup, [110](#)
- kEnableSerial
 - TDCSetup, [110](#)
- kEnableSetCountersOnBunchReset
 - TDCSetup, [110](#)
- kEnableTTLClock
 - TDCSetup, [110](#)
- kEnableTTLControl
 - TDCSetup, [110](#)
- kEnableTTLHit
 - TDCSetup, [110](#)
- kEnableTTLReset
 - TDCSetup, [110](#)
- kEnableTTLSerial
 - TDCSetup, [110](#)
- kEventCountOffset
 - TDCSetup, [110](#)
- kGlobalReset
 - TDCControl, [56](#)
- kIOClockDelay
 - TDCSetup, [110](#)
- kIOClockSource
 - TDCSetup, [110](#)
- kKeepToken
 - TDCSetup, [110](#)
- kLeading
 - TDCSetup, [110](#)
- kLeadingResolution
 - TDCSetup, [110](#)
- kLowPowerMode
 - TDCSetup, [110](#)
- kMaster
 - TDCSetup, [110](#)
- kMatchWindow
 - TDCSetup, [110](#)
- kMaxEventSize
 - TDCSetup, [110](#)
- kModeRC
 - TDCSetup, [111](#)
- kModeRCCompression
 - TDCSetup, [111](#)
- kOffset0
 - TDCSetup, [111](#)
- kPLLControl
 - TDCSetup, [111](#)
- kPLLReset
 - TDCControl, [56](#)
- kRCAdjust0
 - TDCSetup, [111](#)
- kReadoutFIFOSize
 - TDCSetup, [111](#)
- kReadoutSingleCycleSpeed
 - TDCSetup, [111](#)
- kReadoutSpeedSelect
 - TDCSetup, [111](#)
- kRejectCountOffset
 - TDCSetup, [111](#)
- kRejectFIFOFull
 - TDCSetup, [111](#)
- kRollOver
 - TDCSetup, [111](#)
- kSearchWindow
 - TDCSetup, [111](#)
- kSelectBypassInputs
 - TDCSetup, [111](#)
- kSerialClockDelay
 - TDCSetup, [111](#)
- kSerialClockSource
 - TDCSetup, [111](#)
- kSerialDelay
 - TDCSetup, [111](#)
- kSetupParity
 - TDCSetup, [111](#)
- kStrobeSelect
 - TDCSetup, [111](#)
- kTDClId
 - TDCSetup, [111](#)
- kTestInvert
 - TDCSetup, [111](#)
- kTestMode
 - TDCSetup, [111](#)
- kTestSelect
 - TDCSetup, [111](#)
- kTokenDelay
 - TDCSetup, [111](#)
- kTrailing
 - TDCSetup, [111](#)
- kTriggerCountOffset
 - TDCSetup, [111](#)
- kVernierOffset
 - TDCSetup, [111](#)
- kWidthSelect
 - TDCSetup, [111](#)

- L1BufferParityError
 - HPTDC chip control, [11](#)
- LeadingEdge
 - HPTDC chip control, [11](#)
- Listen
 - Socket, [42](#)
- ListenerInfo, [29](#)
 - name, [29](#)
 - type, [29](#)
- MASTER
 - Socket communication objects, [7](#)
- magic
 - file_header_t, [21](#)
- Message, [29](#)
 - ~Message, [31](#)
 - Dump, [31](#)
 - fString, [31](#)
 - GetKey, [31](#)
 - GetString, [31](#)
 - IsFromWeb, [31](#)
 - Message, [31](#)
- Messenger, [31](#)
 - ~Messenger, [33](#)
 - AddClient, [34](#)
 - Broadcast, [34](#)
 - Connect, [35](#)
 - Disconnect, [35](#)
 - DisconnectClient, [36](#)
 - fListenersInfo, [38](#)
 - fNumAttempts, [38](#)
 - fWS, [38](#)
 - GetType, [36](#)
 - Messenger, [33](#)
 - ProcessMessage, [36](#)
 - Receive, [37](#)
 - Send, [37](#)
 - SwitchClientType, [38](#)
- name
 - ListenerInfo, [29](#)
- Object
 - SocketMessage, [50](#)
- OpenFile
 - FPGAHandler, [25](#)
- ParseMessage
 - Client, [17](#)
- PrepareConnection
 - Socket, [43](#)
- ProcessMessage
 - Messenger, [36](#)
- RO_Fixed
 - HPTDC chip control, [12](#)
- RO_pll_80Mbits_s
 - HPTDC chip control, [12](#)
- RSC_10Mbits_s
 - HPTDC chip control, [11](#)
- RSC_1p25Mbits_s
 - HPTDC chip control, [11](#)
- RSC_20Mbits_s
 - HPTDC chip control, [11](#)
- RSC_2p5Mbits_s
 - HPTDC chip control, [11](#)
- RSC_312p5kbits_s
 - HPTDC chip control, [11](#)
- RSC_40Mbits_s
 - HPTDC chip control, [11](#)
- RSC_5Mbits_s
 - HPTDC chip control, [11](#)
- RSC_625kbits_s
 - HPTDC chip control, [11](#)
- ReadBoundaryScanRegister
 - FPGAHandler, [25](#)
- ReadBuffer
 - FPGAHandler, [25](#)
- ReadConfiguration
 - FPGAHandler, [25](#)
- ReadoutFIFOParityError
 - HPTDC chip control, [11](#)
- ReadoutSingleCycleSpeed
 - HPTDC chip control, [11](#)
- ReadoutSpeed
 - HPTDC chip control, [11](#)
- ReadoutStateError
 - HPTDC chip control, [11](#)
- Receive
 - Client, [17](#)
 - Messenger, [37](#)
- run_id
 - file_header_t, [21](#)
- SS_DSSStrobe
 - HPTDC chip control, [12](#)
- SS_LeadingEdge
 - HPTDC chip control, [12](#)
- SS_LeadingTrailingStrobe
 - HPTDC chip control, [12](#)
- SS_NoStrobe
 - HPTDC chip control, [12](#)
- SelectConnections
 - Socket, [43](#)
- Send
 - Client, [17](#)
 - Messenger, [37](#)
- SendConfiguration
 - FPGAHandler, [25](#)
- SendMessage
 - Socket, [43](#)
- Serial_aux_clock
 - HPTDC chip control, [12](#)
- Serial_pll_clock_160
 - HPTDC chip control, [12](#)
- Serial_pll_clock_40
 - HPTDC chip control, [12](#)
- Serial_pll_clock_80
 - HPTDC chip control, [12](#)

- HPTDC chip control, [12](#)
- SerialClockSource
 - HPTDC chip control, [12](#)
- SerialStrobeType
 - HPTDC chip control, [12](#)
- SetAllChannelsOffset
 - TDCSetup, [82](#)
- SetAllTapsDLLAdjustment
 - TDCSetup, [82](#)
- SetBits
 - TDCRegister, [62](#)
- SetBypassInputs
 - TDCSetup, [82](#)
- SetChannelOffset
 - TDCSetup, [83](#)
- SetCoarseCountOffset
 - TDCSetup, [83](#)
- SetConfiguration
 - FPGAHandler, [26](#)
- SetConstantValues
 - TDCSetup, [83](#)
- SetControlParity
 - TDCControl, [55](#)
- SetCoreClockDelay
 - TDCSetup, [84](#)
- SetCoreClockSource
 - TDCSetup, [85](#)
- SetDLLAdjustment
 - TDCSetup, [85](#)
- SetDLLClockDelay
 - TDCSetup, [86](#)
- SetDLLClockSource
 - TDCSetup, [86](#)
- SetDLLControl
 - TDCSetup, [86](#)
- SetDLLMode
 - TDCSetup, [87](#)
- SetDLLReset
 - TDCControl, [55](#)
- SetDeadTime
 - TDCSetup, [85](#)
- SetEdgeResolution
 - TDCSetup, [87](#)
- SetEdgesPairing
 - TDCSetup, [87](#)
- SetEnableAutomaticReject
 - TDCSetup, [88](#)
- SetEnableBytewise
 - TDCSetup, [88](#)
- SetEnableDirectBunchReset
 - TDCSetup, [88](#)
- SetEnableDirectEventReset
 - TDCSetup, [89](#)
- SetEnableDirectTrigger
 - TDCSetup, [89](#)
- SetEnableError
 - TDCSetup, [89](#)
- SetEnableErrorBypass
 - TDCSetup, [90](#)
- SetEnableErrorMark
 - TDCSetup, [90](#)
- SetEnableGlobalHeader
 - TDCSetup, [90](#)
- SetEnableGlobalTrailer
 - TDCSetup, [91](#)
- SetEnableJTAGReadout
 - TDCSetup, [91](#)
- SetEnableLocalHeader
 - TDCSetup, [91](#)
- SetEnableLocalTrailer
 - TDCSetup, [92](#)
- SetEnableMasterResetCode
 - TDCSetup, [92](#)
- SetEnableMasterResetOnEventReset
 - TDCSetup, [92](#)
- SetEnableOverflowDetect
 - TDCSetup, [93](#)
- SetEnablePattern
 - TDCControl, [55](#)
- SetEnableReadoutOccupancy
 - TDCSetup, [93](#)
- SetEnableReadoutSeparator
 - TDCSetup, [93](#)
- SetEnableRelative
 - TDCSetup, [94](#)
- SetEnableResetChannelBufferWhenSeparator
 - TDCSetup, [94](#)
- SetEnableSeparatorOnBunchReset
 - TDCSetup, [94](#)
- SetEnableSeparatorOnEventReset
 - TDCSetup, [95](#)
- SetEnableSerial
 - TDCSetup, [95](#)
- SetEnableSetCountersOnBunchReset
 - TDCSetup, [95](#)
- SetEnableTTLClock
 - TDCSetup, [96](#)
- SetEnableTTLControl
 - TDCSetup, [96](#)
- SetEnableTTLHit
 - TDCSetup, [96](#)
- SetEnableTTLReset
 - TDCSetup, [97](#)
- SetEnableTTLSerial
 - TDCSetup, [97](#)
- SetEventCountOffset
 - TDCSetup, [98](#)
- SetGlobalReset
 - TDCControl, [55](#)
- SetIOClockDelay
 - TDCSetup, [98](#)
- SetIOClockSource
 - TDCSetup, [98](#)
- SetKeepToken
 - TDCSetup, [99](#)
- SetKeyValue

- SocketMessage, [50](#), [51](#)
- SetLeadingMode
 - TDCSetup, [99](#)
- SetLowPowerMode
 - TDCSetup, [99](#)
- SetMaster
 - TDCSetup, [100](#)
- SetMatchWindow
 - TDCSetup, [100](#)
- SetMaxEventSize
 - TDCSetup, [100](#)
- SetModeRC
 - TDCSetup, [101](#)
- SetModeRCCompression
 - TDCSetup, [101](#)
- SetPLLControl
 - TDCSetup, [101](#)
- SetPLLReset
 - TDCControl, [56](#)
- SetPort
 - Socket, [43](#)
- SetRCAdjustment
 - TDCSetup, [102](#)
- SetReadoutFIFOSize
 - TDCSetup, [102](#)
- SetReadoutSingleCycleSpeed
 - TDCSetup, [102](#)
- SetReadoutSpeedSelect
 - TDCSetup, [103](#)
- SetRejectCountOffset
 - TDCSetup, [103](#)
- SetRejectFIFOFull
 - TDCSetup, [103](#)
- SetRollOver
 - TDCSetup, [104](#)
- SetSearchWindow
 - TDCSetup, [104](#)
- SetSerialClockDelay
 - TDCSetup, [104](#)
- SetSerialClockSource
 - TDCSetup, [105](#)
- SetSerialDelay
 - TDCSetup, [105](#)
- SetSetupParity
 - TDCSetup, [105](#)
- SetSocketId
 - Socket, [43](#)
- SetStrobeSelect
 - TDCSetup, [106](#)
- SetTestInvert
 - TDCSetup, [106](#)
- SetTestMode
 - TDCSetup, [106](#)
- SetTokenDelay
 - TDCSetup, [107](#)
- SetTrailingMode
 - TDCSetup, [107](#)
- SetTriggerCountOffset
 - TDCSetup, [107](#)
- SetTriggerMatchingMode
 - TDCSetup, [108](#)
- SetVernierOffset
 - TDCSetup, [108](#)
- SetWidthResolution
 - TDCSetup, [108](#)
- SetWord
 - TDCRegister, [62](#)
- SetupParityError
 - HPTDC chip control, [11](#)
- Socket, [39](#)
 - ~Socket, [41](#)
 - AcceptConnections, [41](#)
 - Bind, [41](#)
 - Configure, [41](#)
 - Create, [42](#)
 - DumpConnected, [42](#)
 - fAddress, [44](#)
 - fBuffer, [44](#)
 - fMaster, [44](#)
 - fPort, [44](#)
 - fReadFds, [44](#)
 - fSocketId, [44](#)
 - fSocketsConnected, [44](#)
 - FetchMessage, [42](#)
 - GetPort, [42](#)
 - GetSocketId, [42](#)
 - GetSocketType, [42](#)
 - IsWebSocket, [42](#)
 - Listen, [42](#)
 - PrepareConnection, [43](#)
 - SelectConnections, [43](#)
 - SendMessage, [43](#)
 - SetPort, [43](#)
 - SetSocketId, [43](#)
 - Socket, [41](#)
 - SocketCollection, [41](#)
 - Start, [43](#)
 - Stop, [44](#)
- Socket communication objects, [7](#)
 - CLIENT, [7](#)
 - DETECTOR, [8](#)
 - INVALID, [7](#)
 - MASTER, [7](#)
 - SocketType, [7](#)
 - WEBSOCKET_CLIENT, [7](#)
- SocketCollection
 - Socket, [41](#)
- SocketMessage, [45](#)
 - ~SocketMessage, [49](#)
 - Dump, [49](#)
 - fMessage, [51](#)
 - GetIntValue, [49](#)
 - GetKey, [49](#)
 - GetString, [50](#)
 - GetValue, [50](#)
 - GetVectorValue, [50](#)

- Object, 50
- SetKeyValue, 50, 51
- SocketMessage, 47–49
- String, 51
- SocketType
 - Socket communication objects, 7
- spill_id
 - file_header_t, 22
- Start
 - Socket, 43
- Stop
 - Socket, 44
- String
 - SocketMessage, 51
- SwitchClientType
 - Messenger, 38
- TDCBoundaryScanRegister, 51
 - ~TDCBoundaryScanRegister, 52
 - TDCBoundaryScanRegister, 52
- TDCControl, 53
 - ~TDCControl, 54
 - Dump, 54
 - EnablePattern, 54
 - kControlParity, 56
 - kDLLReset, 56
 - kEnableChannel, 56
 - kEnablePattern, 56
 - kGlobalReset, 56
 - kPLLReset, 56
 - SetControlParity, 55
 - SetDLLReset, 55
 - SetEnablePattern, 55
 - SetGlobalReset, 55
 - SetPLLReset, 56
 - TDCControl, 54
- TDCEvent, 56
 - ~TDCEvent, 57
 - fWord, 59
 - GetBunchId, 57
 - GetErrorFlags, 57
 - GetEventId, 58
 - GetLeadingTime, 58
 - GetTDCId, 58
 - GetTrailingTime, 58
 - GetType, 59
 - GetWidth, 59
 - GetWordCount, 59
 - TDCEvent, 57
- TDCHeader
 - HPTDC chip control, 11
- TDCRegister, 60
 - ~TDCRegister, 61
 - bit, 61
 - DumpRegister, 61
 - fNumWords, 62
 - fWord, 62
 - GetBits, 61
 - GetNumWords, 61
 - GetWord, 61
 - SetBits, 62
 - SetWord, 62
 - TDCRegister, 61
 - word_t, 61
- TDCSetup, 62
 - ~TDCSetup, 71
 - Dump, 71
 - GetChannelOffset, 72
 - GetCoarseCountOffset, 72
 - GetDLLAdjustment, 73
 - GetDeadTime, 73
 - GetEdgeResolution, 73
 - GetEdgesPairing, 74
 - GetEnableError, 74
 - GetEnableErrorBypass, 74
 - GetEnableErrorMark, 75
 - GetEnableJTAGReadout, 75
 - GetEnableReadoutOccupancy, 75
 - GetEnableReadoutSeparator, 76
 - GetEnableSerial, 76
 - GetLeadingMode, 76
 - GetMatchWindow, 77
 - GetMaxEventSize, 77
 - GetRCAdjustment, 77
 - GetReadoutFIFOSize, 78
 - GetRejectCountOffset, 78
 - GetRejectFIFOFull, 78
 - GetSearchWindow, 79
 - GetSetupParity, 79
 - GetTestInvert, 79
 - GetTestMode, 80
 - GetTrailingMode, 80
 - GetTriggerCountOffset, 80
 - GetTriggerLatency, 81
 - GetTriggerMatchingMode, 81
 - GetVernierOffset, 81
 - GetWidthResolution, 82
 - kCoarseCountOffset, 109
 - kCoreClockDelay, 109
 - kCoreClockSource, 109
 - kDLLClockDelay, 109
 - kDLLClockSource, 109
 - kDLLControl, 109
 - kDLLMode, 109
 - kDLLTapAdjust0, 109
 - kDeadTime, 109
 - kEnableAutomaticReject, 109
 - kEnableBytewise, 109
 - kEnableDirectBunchReset, 109
 - kEnableDirectEventReset, 109
 - kEnableDirectTrigger, 109
 - kEnableError, 109
 - kEnableErrorBypass, 109
 - kEnableErrorMark, 109
 - kEnableGlobalHeader, 109
 - kEnableGlobalTrailer, 109
 - kEnableJTAGReadout, 109

- kEnableLocalHeader, 109
- kEnableLocalTrailer, 109
- kEnableMasterResetCode, 110
- kEnableMasterResetOnEventReset, 110
- kEnableMatching, 110
- kEnableOverflowDetect, 110
- kEnablePair, 110
- kEnableReadoutOccupancy, 110
- kEnableReadoutSeparator, 110
- kEnableRelative, 110
- kEnableResetChannelBufferWhenSeparator, 110
- kEnableSeparatorOnBunchReset, 110
- kEnableSeparatorOnEventReset, 110
- kEnableSerial, 110
- kEnableSetCountersOnBunchReset, 110
- kEnableTTLClock, 110
- kEnableTTLControl, 110
- kEnableTTLHit, 110
- kEnableTTLReset, 110
- kEnableTTLSerial, 110
- kEventCountOffset, 110
- kIOClockDelay, 110
- kIOClockSource, 110
- kKeepToken, 110
- kLeading, 110
- kLeadingResolution, 110
- kLowPowerMode, 110
- kMaster, 110
- kMatchWindow, 110
- kMaxEventSize, 110
- kModeRC, 111
- kModeRCCompression, 111
- kOffset0, 111
- kPLLControl, 111
- kRCAdjust0, 111
- kReadoutFIFOSize, 111
- kReadoutSingleCycleSpeed, 111
- kReadoutSpeedSelect, 111
- kRejectCountOffset, 111
- kRejectFIFOFull, 111
- kRollOver, 111
- kSearchWindow, 111
- kSelectBypassInputs, 111
- kSerialClockDelay, 111
- kSerialClockSource, 111
- kSerialDelay, 111
- kSetupParity, 111
- kStrobeSelect, 111
- kTDCId, 111
- kTestInvert, 111
- kTestMode, 111
- kTestSelect, 111
- kTokenDelay, 111
- kTrailing, 111
- kTriggerCountOffset, 111
- kVernierOffset, 111
- kWidthSelect, 111
- SetAllChannelsOffset, 82
- SetAllTapsDLLAdjustment, 82
- SetBypassInputs, 82
- SetChannelOffset, 83
- SetCoarseCountOffset, 83
- SetConstantValues, 83
- SetCoreClockDelay, 84
- SetCoreClockSource, 85
- SetDLLAdjustment, 85
- SetDLLClockDelay, 86
- SetDLLClockSource, 86
- SetDLLControl, 86
- SetDLLMode, 87
- SetDeadTime, 85
- SetEdgeResolution, 87
- SetEdgesPairing, 87
- SetEnableAutomaticReject, 88
- SetEnableBytewise, 88
- SetEnableDirectBunchReset, 88
- SetEnableDirectEventReset, 89
- SetEnableDirectTrigger, 89
- SetEnableError, 89
- SetEnableErrorBypass, 90
- SetEnableErrorMark, 90
- SetEnableGlobalHeader, 90
- SetEnableGlobalTrailer, 91
- SetEnableJTAGReadout, 91
- SetEnableLocalHeader, 91
- SetEnableLocalTrailer, 92
- SetEnableMasterResetCode, 92
- SetEnableMasterResetOnEventReset, 92
- SetEnableOverflowDetect, 93
- SetEnableReadoutOccupancy, 93
- SetEnableReadoutSeparator, 93
- SetEnableRelative, 94
- SetEnableResetChannelBufferWhenSeparator, 94
- SetEnableSeparatorOnBunchReset, 94
- SetEnableSeparatorOnEventReset, 95
- SetEnableSerial, 95
- SetEnableSetCountersOnBunchReset, 95
- SetEnableTTLClock, 96
- SetEnableTTLControl, 96
- SetEnableTTLHit, 96
- SetEnableTTLReset, 97
- SetEnableTTLSerial, 97
- SetEventCountOffset, 98
- SetIOClockDelay, 98
- SetIOClockSource, 98
- SetKeepToken, 99
- SetLeadingMode, 99
- SetLowPowerMode, 99
- SetMaster, 100
- SetMatchWindow, 100
- SetMaxEventSize, 100
- SetModeRC, 101
- SetModeRCCompression, 101
- SetPLLControl, 101
- SetRCAdjustment, 102
- SetReadoutFIFOSize, 102

- SetReadoutSingleCycleSpeed, [102](#)
- SetReadoutSpeedSelect, [103](#)
- SetRejectCountOffset, [103](#)
- SetRejectFIFOFull, [103](#)
- SetRollOver, [104](#)
- SetSearchWindow, [104](#)
- SetSerialClockDelay, [104](#)
- SetSerialClockSource, [105](#)
- SetSerialDelay, [105](#)
- SetSetupParity, [105](#)
- SetStrobeSelect, [106](#)
- SetTestInvert, [106](#)
- SetTestMode, [106](#)
- SetTokenDelay, [107](#)
- SetTrailingMode, [107](#)
- SetTriggerCountOffset, [107](#)
- SetTriggerMatchingMode, [108](#)
- SetVernierOffset, [108](#)
- SetWidthResolution, [108](#)
- TDCSetup, [70](#)
- TDCTrailer
 - HPTDC chip control, [11](#)
- TrailingEdge
 - HPTDC chip control, [11](#)
- TriggerFIFOParityError
 - HPTDC chip control, [11](#)
- TriggerMatchingError
 - HPTDC chip control, [11](#)
- Type
 - Exception, [20](#)
- type
 - ListenerInfo, [29](#)
- TypeString
 - Exception, [20](#)
- USBHandler, [112](#)
 - ~USBHandler, [113](#)
 - DumpDevice, [113](#)
 - fDevice, [113](#)
 - fHandle, [113](#)
 - Fetch, [113](#)
 - Init, [113](#)
 - USBHandler, [113](#)
 - Write, [113](#)
- VernierError
 - HPTDC chip control, [11](#)
- W_100ns
 - HPTDC chip control, [12](#)
- W_100ps
 - HPTDC chip control, [12](#)
- W_12p5ns
 - HPTDC chip control, [12](#)
- W_1p6ns
 - HPTDC chip control, [12](#)
- W_200ns
 - HPTDC chip control, [12](#)
- W_200ps
 - HPTDC chip control, [12](#)
- W_25ns
 - HPTDC chip control, [12](#)
- W_3p2ns
 - HPTDC chip control, [12](#)
- W_400ns
 - HPTDC chip control, [12](#)
- W_400ps
 - HPTDC chip control, [12](#)
- W_50ns
 - HPTDC chip control, [12](#)
- W_6p25ns
 - HPTDC chip control, [12](#)
- W_800ns
 - HPTDC chip control, [12](#)
- W_800ps
 - HPTDC chip control, [12](#)
- WEBSOCKET_CLIENT
 - Socket communication objects, [7](#)
- WidthResolution
 - HPTDC chip control, [12](#)
- word_t
 - TDCRegister, [61](#)
- Write
 - USBHandler, [113](#)