

## 2015 Test beam Run Control

Generated by Doxygen 1.8.9.1

Mon Apr 27 2015 10:04:20



# Contents

<b>1</b>	<b>Module Index</b>	<b>1</b>
1.1	Modules . . . . .	1
<b>2</b>	<b>Hierarchical Index</b>	<b>3</b>
2.1	Class Hierarchy . . . . .	3
<b>3</b>	<b>Data Structure Index</b>	<b>5</b>
3.1	Data Structures . . . . .	5
<b>4</b>	<b>Module Documentation</b>	<b>7</b>
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 . . . . .	11
4.2.2.5	EdgeResolution . . . . .	11
4.2.2.6	EnabledError . . . . .	11
4.2.2.7	EnablePattern . . . . .	11
4.2.2.8	EventType . . . . .	11
4.2.2.9	IOClockSource . . . . .	12
4.2.2.10	ReadoutSingleCycleSpeed . . . . .	12
4.2.2.11	ReadoutSpeed . . . . .	12
4.2.2.12	RegisterName . . . . .	12
4.2.2.13	SerialClockSource . . . . .	12
4.2.2.14	SerialStrobeType . . . . .	13
4.2.2.15	WidthResolution . . . . .	13

<b>5</b>	<b>Data Structure Documentation</b>	<b>15</b>
5.1	Client Class Reference	15
5.1.1	Detailed Description	16
5.1.2	Constructor & Destructor Documentation	17
5.1.2.1	Client	17
5.1.2.2	Client	17
5.1.2.3	~Client	17
5.1.3	Member Function Documentation	17
5.1.3.1	Announce	17
5.1.3.2	Connect	18
5.1.3.3	Disconnect	18
5.1.3.4	GetType	19
5.1.3.5	ParseMessage	19
5.1.3.6	Receive	19
5.1.3.7	Send	20
5.1.4	Field Documentation	20
5.1.4.1	fClientId	20
5.1.4.2	flsConnected	20
5.2	Exception Class Reference	20
5.2.1	Detailed Description	21
5.2.2	Constructor & Destructor Documentation	21
5.2.2.1	Exception	21
5.2.2.2	Exception	21
5.2.2.3	~Exception	21
5.2.3	Member Function Documentation	21
5.2.3.1	Description	21
5.2.3.2	Dump	22
5.2.3.3	ErrorNumber	22
5.2.3.4	From	22
5.2.3.5	Type	22
5.2.3.6	TypeString	22
5.2.4	Field Documentation	22
5.2.4.1	fDescription	22
5.2.4.2	fErrorNumber	22
5.2.4.3	fFrom	22
5.2.4.4	fType	22
5.3	file_header_t Struct Reference	23
5.3.1	Detailed Description	23
5.3.2	Field Documentation	23
5.3.2.1	config	23

5.3.2.2	magic	23
5.3.2.3	run_id	24
5.3.2.4	spill_id	24
5.4	FPGAHandler Class Reference	24
5.4.1	Detailed Description	25
5.4.2	Constructor & Destructor Documentation	26
5.4.2.1	FPGAHandler	26
5.4.2.2	~FPGAHandler	26
5.4.3	Member Function Documentation	26
5.4.3.1	CloseFile	26
5.4.3.2	ErrorState	26
5.4.3.3	GetConfiguration	26
5.4.3.4	GetFilename	27
5.4.3.5	GetType	27
5.4.3.6	OpenFile	27
5.4.3.7	ReadBoundaryScanRegister	27
5.4.3.8	ReadBuffer	27
5.4.3.9	ReadConfiguration	27
5.4.3.10	ReadRegister	27
5.4.3.11	SendConfiguration	28
5.4.3.12	SetConfiguration	28
5.4.3.13	SetRegister	28
5.4.4	Field Documentation	29
5.4.4.1	fFilename	29
5.4.4.2	flsFileOpen	29
5.4.4.3	flsTDCInReadout	29
5.4.4.4	fOutput	29
5.4.4.5	fTDCBSR	29
5.4.4.6	fTDCControl	29
5.4.4.7	fTDCSetup	29
5.4.4.8	fTDCStatus	29
5.5	HTTPMessage Class Reference	29
5.5.1	Detailed Description	30
5.5.2	Constructor & Destructor Documentation	30
5.5.2.1	HTTPMessage	31
5.5.2.2	HTTPMessage	31
5.5.3	Member Function Documentation	31
5.5.3.1	Decode	31
5.5.3.2	Dump	31
5.5.3.3	Encode	31

5.5.3.4	GetKey	31
5.5.4	Field Documentation	31
5.5.4.1	fOriginalString	31
5.5.4.2	fWS	31
5.6	ListenerInfo Struct Reference	32
5.6.1	Detailed Description	32
5.6.2	Field Documentation	32
5.6.2.1	name	32
5.6.2.2	type	32
5.7	Message Class Reference	32
5.7.1	Detailed Description	33
5.7.2	Constructor & Destructor Documentation	33
5.7.2.1	Message	33
5.7.2.2	Message	33
5.7.2.3	Message	33
5.7.2.4	~Message	34
5.7.3	Member Function Documentation	34
5.7.3.1	Dump	34
5.7.3.2	GetKey	34
5.7.3.3	GetString	34
5.7.3.4	IsFromWeb	34
5.7.4	Field Documentation	34
5.7.4.1	fString	34
5.8	Messenger Class Reference	34
5.8.1	Detailed Description	36
5.8.2	Constructor & Destructor Documentation	36
5.8.2.1	Messenger	36
5.8.2.2	Messenger	36
5.8.2.3	~Messenger	36
5.8.3	Member Function Documentation	37
5.8.3.1	AddClient	37
5.8.3.2	Broadcast	37
5.8.3.3	Connect	38
5.8.3.4	Disconnect	38
5.8.3.5	DisconnectClient	38
5.8.3.6	GetType	39
5.8.3.7	ProcessMessage	39
5.8.3.8	Receive	39
5.8.3.9	Send	40
5.8.3.10	SwitchClientType	41

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

5.10.1 Detailed Description . . . . .	49
5.10.2 Constructor & Destructor Documentation . . . . .	49
5.10.2.1 SocketMessage . . . . .	49
5.10.2.2 SocketMessage . . . . .	49
5.10.2.3 SocketMessage . . . . .	50
5.10.2.4 SocketMessage . . . . .	50
5.10.2.5 SocketMessage . . . . .	50
5.10.2.6 SocketMessage . . . . .	50
5.10.2.7 SocketMessage . . . . .	51
5.10.2.8 SocketMessage . . . . .	51
5.10.2.9 SocketMessage . . . . .	51
5.10.2.10 SocketMessage . . . . .	51
5.10.2.11 SocketMessage . . . . .	52
5.10.2.12 ~SocketMessage . . . . .	52
5.10.3 Member Function Documentation . . . . .	52
5.10.3.1 Dump . . . . .	52
5.10.3.2 GetIntValue . . . . .	52
5.10.3.3 GetKey . . . . .	52
5.10.3.4 GetString . . . . .	52
5.10.3.5 GetValue . . . . .	52
5.10.3.6 GetVectorValue . . . . .	52
5.10.3.7 Object . . . . .	53
5.10.3.8 SetKeyValue . . . . .	53
5.10.3.9 SetKeyValue . . . . .	53
5.10.3.10 SetKeyValue . . . . .	53
5.10.3.11 SetKeyValue . . . . .	54
5.10.3.12 String . . . . .	54
5.10.4 Field Documentation . . . . .	54
5.10.4.1 fMessage . . . . .	54
5.11 TDCBoundaryScanRegister Class Reference . . . . .	54
5.11.1 Detailed Description . . . . .	55
5.11.2 Constructor & Destructor Documentation . . . . .	55
5.11.2.1 TDCBoundaryScanRegister . . . . .	55
5.11.2.2 TDCBoundaryScanRegister . . . . .	56
5.11.3 Member Function Documentation . . . . .	56
5.11.3.1 SetConstantValues . . . . .	56
5.12 TDCCControl Class Reference . . . . .	56
5.12.1 Detailed Description . . . . .	58
5.12.2 Constructor & Destructor Documentation . . . . .	58
5.12.2.1 TDCCControl . . . . .	58



5.12.2.2	TDCCControl	58
5.12.3	Member Function Documentation	58
5.12.3.1	DisableAllChannels	58
5.12.3.2	DisableChannel	59
5.12.3.3	Dump	59
5.12.3.4	EnableAllChannels	59
5.12.3.5	EnableChannel	59
5.12.3.6	GetDLLReset	60
5.12.3.7	GetEnablePattern	60
5.12.3.8	GetGlobalReset	60
5.12.3.9	GetPLLReset	61
5.12.3.10	SetConstantValues	61
5.12.3.11	SetControlParity	61
5.12.3.12	SetDLLReset	61
5.12.3.13	SetEnablePattern	62
5.12.3.14	SetGlobalReset	62
5.12.3.15	SetPLLReset	62
5.12.4	Field Documentation	62
5.12.4.1	kControlParity	62
5.12.4.2	kDLLReset	62
5.12.4.3	kEnableChannel	62
5.12.4.4	kEnablePattern	62
5.12.4.5	kGlobalReset	62
5.12.4.6	kPLLReset	63
5.13	TDCEvent Class Reference	63
5.13.1	Detailed Description	64
5.13.2	Constructor & Destructor Documentation	64
5.13.2.1	TDCEvent	64
5.13.2.2	~TDCEvent	64
5.13.3	Member Function Documentation	64
5.13.3.1	GetBunchId	64
5.13.3.2	GetErrorFlags	64
5.13.3.3	GetEventId	65
5.13.3.4	GetLeadingTime	65
5.13.3.5	GetTDCId	65
5.13.3.6	GetTrailingTime	65
5.13.3.7	GetType	66
5.13.3.8	GetWidth	66
5.13.3.9	GetWordCount	66
5.13.4	Field Documentation	66

5.13.4.1	fWord	66
5.14	TDCRegister Class Reference	66
5.14.1	Member Typedef Documentation	68
5.14.1.1	bit	68
5.14.1.2	word_t	68
5.14.2	Constructor & Destructor Documentation	68
5.14.2.1	TDCRegister	68
5.14.2.2	TDCRegister	68
5.14.2.3	~TDCRegister	68
5.14.3	Member Function Documentation	68
5.14.3.1	Clear	68
5.14.3.2	DumpRegister	68
5.14.3.3	GetBits	69
5.14.3.4	GetNumWords	69
5.14.3.5	GetWord	69
5.14.3.6	SetBits	69
5.14.3.7	SetConstantValues	69
5.14.3.8	SetWord	69
5.14.4	Field Documentation	69
5.14.4.1	fNumWords	69
5.14.4.2	fWord	69
5.14.4.3	fWordSize	70
5.15	TDCSetup Class Reference	70
5.15.1	Detailed Description	77
5.15.2	Constructor & Destructor Documentation	77
5.15.2.1	TDCSetup	78
5.15.2.2	TDCSetup	79
5.15.3	Member Function Documentation	79
5.15.3.1	Dump	80
5.15.3.2	GetChannelOffset	80
5.15.3.3	GetCoarseCountOffset	80
5.15.3.4	GetDeadTime	81
5.15.3.5	GetDLLAdjustment	81
5.15.3.6	GetEdgeResolution	82
5.15.3.7	GetEdgesPairing	82
5.15.3.8	GetEnableError	82
5.15.3.9	GetEnableErrorBypass	83
5.15.3.10	GetEnableErrorMark	83
5.15.3.11	GetEnableJTAGReadout	83
5.15.3.12	GetEnableReadoutOccupancy	84

5.15.3.13 GetEnableReadoutSeparator . . . . .	84
5.15.3.14 GetEnableSerial . . . . .	84
5.15.3.15 GetLeadingMode . . . . .	84
5.15.3.16 GetMatchWindow . . . . .	85
5.15.3.17 GetMaxEventSize . . . . .	85
5.15.3.18 GetRCAdjustment . . . . .	85
5.15.3.19 GetReadoutFIFOSize . . . . .	86
5.15.3.20 GetRejectCountOffset . . . . .	86
5.15.3.21 GetRejectFIFOFull . . . . .	86
5.15.3.22 GetSearchWindow . . . . .	87
5.15.3.23 GetSetupParity . . . . .	87
5.15.3.24 GetTestInvert . . . . .	88
5.15.3.25 GetTestMode . . . . .	88
5.15.3.26 GetTrailingMode . . . . .	88
5.15.3.27 GetTriggerCountOffset . . . . .	88
5.15.3.28 GetTriggerLatency . . . . .	89
5.15.3.29 GetTriggerMatchingMode . . . . .	89
5.15.3.30 GetVernierOffset . . . . .	89
5.15.3.31 GetWidthResolution . . . . .	90
5.15.3.32 SetAllChannelsOffset . . . . .	90
5.15.3.33 SetAllTapsDLLAdjustment . . . . .	90
5.15.3.34 SetBypassInputs . . . . .	91
5.15.3.35 SetChannelOffset . . . . .	91
5.15.3.36 SetCoarseCountOffset . . . . .	91
5.15.3.37 SetConstantValues . . . . .	91
5.15.3.38 SetCoreClockDelay . . . . .	92
5.15.3.39 SetCoreClockSource . . . . .	93
5.15.3.40 SetDeadTime . . . . .	93
5.15.3.41 SetDLLAdjustment . . . . .	93
5.15.3.42 SetDLLClockDelay . . . . .	94
5.15.3.43 SetDLLClockSource . . . . .	94
5.15.3.44 SetDLLControl . . . . .	94
5.15.3.45 SetDLLMode . . . . .	95
5.15.3.46 SetEdgeResolution . . . . .	95
5.15.3.47 SetEdgesPairing . . . . .	95
5.15.3.48 SetEnableAutomaticReject . . . . .	96
5.15.3.49 SetEnableBytewise . . . . .	96
5.15.3.50 SetEnableDirectBunchReset . . . . .	96
5.15.3.51 SetEnableDirectEventReset . . . . .	97
5.15.3.52 SetEnableDirectTrigger . . . . .	97

5.15.3.53 SetEnableError . . . . .	97
5.15.3.54 SetEnableErrorBypass . . . . .	98
5.15.3.55 SetEnableErrorMark . . . . .	98
5.15.3.56 SetEnableGlobalHeader . . . . .	98
5.15.3.57 SetEnableGlobalTrailer . . . . .	99
5.15.3.58 SetEnableJTAGReadout . . . . .	99
5.15.3.59 SetEnableLocalHeader . . . . .	99
5.15.3.60 SetEnableLocalTrailer . . . . .	100
5.15.3.61 SetEnableMasterResetCode . . . . .	100
5.15.3.62 SetEnableMasterResetOnEventReset . . . . .	100
5.15.3.63 SetEnableOverflowDetect . . . . .	101
5.15.3.64 SetEnableReadoutOccupancy . . . . .	101
5.15.3.65 SetEnableReadoutSeparator . . . . .	101
5.15.3.66 SetEnableRelative . . . . .	102
5.15.3.67 SetEnableResetChannelBufferWhenSeparator . . . . .	102
5.15.3.68 SetEnableSeparatorOnBunchReset . . . . .	102
5.15.3.69 SetEnableSeparatorOnEventReset . . . . .	103
5.15.3.70 SetEnableSerial . . . . .	103
5.15.3.71 SetEnableSetCountersOnBunchReset . . . . .	103
5.15.3.72 SetEnableTTLClock . . . . .	104
5.15.3.73 SetEnableTTLControl . . . . .	104
5.15.3.74 SetEnableTTLHit . . . . .	105
5.15.3.75 SetEnableTTLReset . . . . .	105
5.15.3.76 SetEnableTTLSerial . . . . .	105
5.15.3.77 SetEventCountOffset . . . . .	106
5.15.3.78 SetIOClockDelay . . . . .	106
5.15.3.79 SetIOClockSource . . . . .	106
5.15.3.80 SetKeepToken . . . . .	107
5.15.3.81 SetLeadingMode . . . . .	107
5.15.3.82 SetLowPowerMode . . . . .	107
5.15.3.83 SetMaster . . . . .	108
5.15.3.84 SetMatchWindow . . . . .	108
5.15.3.85 SetMaxEventSize . . . . .	108
5.15.3.86 SetModeRC . . . . .	109
5.15.3.87 SetModeRCCompression . . . . .	109
5.15.3.88 SetPLLControl . . . . .	109
5.15.3.89 SetRCAdjustment . . . . .	110
5.15.3.90 SetReadoutFIFOSize . . . . .	110
5.15.3.91 SetReadoutSingleCycleSpeed . . . . .	110
5.15.3.92 SetReadoutSpeedSelect . . . . .	111

5.15.3.93 SetRejectCountOffset . . . . .	111
5.15.3.94 SetRejectFIFOFull . . . . .	112
5.15.3.95 SetRollOver . . . . .	112
5.15.3.96 SetSearchWindow . . . . .	112
5.15.3.97 SetSerialClockDelay . . . . .	112
5.15.3.98 SetSerialClockSource . . . . .	113
5.15.3.99 SetSerialDelay . . . . .	113
5.15.3.100SetSetupParity . . . . .	113
5.15.3.101SetStrobeSelect . . . . .	114
5.15.3.102SetTestInvert . . . . .	114
5.15.3.103SetTestMode . . . . .	114
5.15.3.104SetTokenDelay . . . . .	115
5.15.3.105SetTrailingMode . . . . .	115
5.15.3.106SetTriggerCountOffset . . . . .	115
5.15.3.107SetTriggerMatchingMode . . . . .	116
5.15.3.108SetVernierOffset . . . . .	116
5.15.3.109SetWidthResolution . . . . .	116
5.15.4 Field Documentation . . . . .	117
5.15.4.1 kCoarseCountOffset . . . . .	117
5.15.4.2 kCoreClockDelay . . . . .	117
5.15.4.3 kCoreClockSource . . . . .	117
5.15.4.4 kDeadTime . . . . .	117
5.15.4.5 kDLLClockDelay . . . . .	117
5.15.4.6 kDLLClockSource . . . . .	117
5.15.4.7 kDLLControl . . . . .	117
5.15.4.8 kDLLMode . . . . .	117
5.15.4.9 kDLLTapAdjust0 . . . . .	117
5.15.4.10 kEnableAutomaticReject . . . . .	117
5.15.4.11 kEnableBytewise . . . . .	117
5.15.4.12 kEnableDirectBunchReset . . . . .	117
5.15.4.13 kEnableDirectEventReset . . . . .	117
5.15.4.14 kEnableDirectTrigger . . . . .	117
5.15.4.15 kEnableError . . . . .	117
5.15.4.16 kEnableErrorBypass . . . . .	117
5.15.4.17 kEnableErrorMark . . . . .	117
5.15.4.18 kEnableGlobalHeader . . . . .	117
5.15.4.19 kEnableGlobalTrailer . . . . .	117
5.15.4.20 kEnableJTAGReadout . . . . .	117
5.15.4.21 kEnableLocalHeader . . . . .	117
5.15.4.22 kEnableLocalTrailer . . . . .	118

5.15.4.23 kEnableMasterResetCode . . . . .	118
5.15.4.24 kEnableMasterResetOnEventReset . . . . .	118
5.15.4.25 kEnableMatching . . . . .	118
5.15.4.26 kEnableOverflowDetect . . . . .	118
5.15.4.27 kEnablePair . . . . .	118
5.15.4.28 kEnableReadoutOccupancy . . . . .	118
5.15.4.29 kEnableReadoutSeparator . . . . .	118
5.15.4.30 kEnableRelative . . . . .	118
5.15.4.31 kEnableResetChannelBufferWhenSeparator . . . . .	118
5.15.4.32 kEnableSeparatorOnBunchReset . . . . .	118
5.15.4.33 kEnableSeparatorOnEventReset . . . . .	118
5.15.4.34 kEnableSerial . . . . .	118
5.15.4.35 kEnableSetCountersOnBunchReset . . . . .	118
5.15.4.36 kEnableTTLClock . . . . .	118
5.15.4.37 kEnableTTLControl . . . . .	118
5.15.4.38 kEnableTTLHit . . . . .	118
5.15.4.39 kEnableTTLReset . . . . .	118
5.15.4.40 kEnableTTLSerial . . . . .	118
5.15.4.41 kEventCountOffset . . . . .	118
5.15.4.42 kIOClockDelay . . . . .	118
5.15.4.43 kIOClockSource . . . . .	118
5.15.4.44 kKeepToken . . . . .	118
5.15.4.45 kLeading . . . . .	118
5.15.4.46 kLeadingResolution . . . . .	118
5.15.4.47 kLowPowerMode . . . . .	118
5.15.4.48 kMaster . . . . .	118
5.15.4.49 kMatchWindow . . . . .	118
5.15.4.50 kMaxEventSize . . . . .	119
5.15.4.51 kModeRC . . . . .	119
5.15.4.52 kModeRCCompression . . . . .	119
5.15.4.53 kOffset0 . . . . .	119
5.15.4.54 kPLLControl . . . . .	119
5.15.4.55 kRCAdjust0 . . . . .	119
5.15.4.56 kReadoutFIFOSize . . . . .	119
5.15.4.57 kReadoutSingleCycleSpeed . . . . .	119
5.15.4.58 kReadoutSpeedSelect . . . . .	119
5.15.4.59 kRejectCountOffset . . . . .	119
5.15.4.60 kRejectFIFOFull . . . . .	119
5.15.4.61 kRollOver . . . . .	119
5.15.4.62 kSearchWindow . . . . .	119

5.15.4.63 kSelectBypassInputs . . . . .	119
5.15.4.64 kSerialClockDelay . . . . .	119
5.15.4.65 kSerialClockSource . . . . .	119
5.15.4.66 kSerialDelay . . . . .	119
5.15.4.67 kSetupParity . . . . .	119
5.15.4.68 kStrobeSelect . . . . .	119
5.15.4.69 kTDCId . . . . .	119
5.15.4.70 kTestInvert . . . . .	119
5.15.4.71 kTestMode . . . . .	119
5.15.4.72 kTestSelect . . . . .	119
5.15.4.73 kTokenDelay . . . . .	119
5.15.4.74 kTrailing . . . . .	119
5.15.4.75 kTriggerCountOffset . . . . .	119
5.15.4.76 kVernierOffset . . . . .	119
5.15.4.77 kWidthSelect . . . . .	120
5.16 TDCStatus Class Reference . . . . .	120
5.16.1 Detailed Description . . . . .	121
5.16.2 Constructor & Destructor Documentation . . . . .	121
5.16.2.1 TDCStatus . . . . .	121
5.16.2.2 TDCStatus . . . . .	121
5.16.3 Member Function Documentation . . . . .	121
5.16.3.1 SetConstantValues . . . . .	122
5.16.4 Field Documentation . . . . .	122
5.16.4.1 kDLLLock . . . . .	122
5.16.4.2 kError . . . . .	122
5.16.4.3 kHaveToken . . . . .	122
5.16.4.4 kL1Occupancy . . . . .	122
5.16.4.5 kReadoutFIFOEmpty . . . . .	122
5.16.4.6 kReadoutFIFOFull . . . . .	122
5.16.4.7 kReadoutFIFOOccupancy . . . . .	122
5.16.4.8 kTriggerFIFOEmpty . . . . .	122
5.16.4.9 kTriggerFIFOFull . . . . .	122
5.16.4.10 kTriggerFIFOOccupancy . . . . .	122
5.17 USBHandler Class Reference . . . . .	122
5.17.1 Detailed Description . . . . .	123
5.17.2 Constructor & Destructor Documentation . . . . .	123
5.17.2.1 USBHandler . . . . .	123
5.17.2.2 ~USBHandler . . . . .	123
5.17.3 Member Function Documentation . . . . .	123
5.17.3.1 DumpDevice . . . . .	123

---

5.17.3.2	Fetch . . . . .	123
5.17.3.3	Init . . . . .	123
5.17.3.4	Write . . . . .	124
5.17.4	Field Documentation . . . . .	124
5.17.4.1	fDevice . . . . .	124
5.17.4.2	fHandle . . . . .	124
<b>Index</b>		<b>125</b>



# 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 . . . . .	20
file_header_t . . . . .	23
ListenerInfo . . . . .	32
Message . . . . .	32
HTTPMessage . . . . .	29
SocketMessage . . . . .	47
Socket . . . . .	41
Client . . . . .	15
FPGAHandler . . . . .	24
Messenger . . . . .	34
TDCEvent . . . . .	63
TDCRegister . . . . .	66
TDCBoundaryScanRegister . . . . .	54
TDCControl . . . . .	56
TDCSetup . . . . .	70
TDCStatus . . . . .	120
USBHandler . . . . .	122
FPGAHandler . . . . .	24



## Chapter 3

# Data Structure Index

### 3.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">Client</a>	Base client object for the socket . . . . .	15
<a href="#">Exception</a>	A simple exception handler . . . . .	20
<a href="#">file_header_t</a>	Header to the output files . . . . .	23
<a href="#">FPGAHandler</a>	Driver for timing detectors' FPGA readout . . . . .	24
<a href="#">HTTPMessage</a>	<a href="#">Message</a> to be transmitted through a WebSocket protocol . . . . .	29
<a href="#">ListenerInfo</a>	Information on a socket's listener . . . . .	32
<a href="#">Message</a>	Base socket message type . . . . .	32
<a href="#">Messenger</a>	Base master object for the socket . . . . .	34
<a href="#">Socket</a>	Base socket object from which clients/master from a socket inherit . . . . .	41
<a href="#">SocketMessage</a>	Socket-passed message type . . . . .	47
<a href="#">TDCBoundaryScanRegister</a>	. . . . .	54
<a href="#">TDCControl</a>	Control word to be sent to the HPTDC chip . . . . .	56
<a href="#">TDCEvent</a>	HPTDC event parser . . . . .	63
<a href="#">TDCRegister</a>	. . . . .	66
<a href="#">TDCSetup</a>	Setup word to be sent to the HPTDC chip . . . . .	70
<a href="#">TDCStatus</a>	. . . . .	120
<a href="#">USBHandler</a>	Generic USB communication handler . . . . .	122



## 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

General register object to interact with a HPTDC chip.

### Data Structures

- class [TDCBoundaryScanRegister](#)
- class [TDCControl](#)  
*Control word to be sent to the HPTDC chip.*
- class [TDCEvent](#)  
*HPTDC event parser.*
- class [TDCSetup](#)  
*Setup word to be sent to the HPTDC chip.*
- class [TDCStatus](#)

### Enumerations

- enum [TDCControl::EnablePattern](#)
- enum [TDCControl::RegisterName](#) { [TDCControl::R\\_EnablePattern](#), [TDCControl::R\\_GlobalReset](#), [TDCControl::R\\_DLLReset](#), [TDCControl::R\\_PLLReset](#) }
- 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\\_80Mbps\\_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

General register object to interact with a HPTDC chip.

Author

Laurent Forthomme [laurent.forthomme@cern.ch](mailto:laurent.forthomme@cern.ch)

Date

24 Apr 2015

#### 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 TDCControl::EnablePattern

## 4.2.2.8 enum TDCEvent::EventType

Enumerator

*Invalid*  
*GroupHeader*  
*GroupTrailer*  
*TDCHeader*  
*TDCTrailer*

*LeadingEdge*  
*TrailingEdge*  
*Error*  
*Debug*

#### 4.2.2.9 enum TDCSetup::IOClockSource

Enumerator

*IO\_clock\_40*  
*IO\_pll\_clock\_80*  
*IO\_pll\_clock\_160*  
*IO\_aux\_clock*

#### 4.2.2.10 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*

#### 4.2.2.11 enum TDCSetup::ReadoutSpeed

Enumerator

*RO\_Fixed*  
*RO\_pll\_80Mbits\_s*

#### 4.2.2.12 enum TDCCControl::RegisterName

Enumerator

*R\_EnablePattern*  
*R\_GlobalReset*  
*R\_DLLReset*  
*R\_PLLReset*

#### 4.2.2.13 enum TDCSetup::SerialClockSource

Enumerator

*Serial\_pll\_clock\_80*  
*Serial\_pll\_clock\_160*  
*Serial\_pll\_clock\_40*  
*Serial\_aux\_clock*

## 4.2.2.14 enum TDCSetup::SerialStrobeType

Enumerator

*SS\_NoStrobe**SS\_DSStrobe**SS\_LeadingTrailingStrobe**SS\_LeadingEdge*

## 4.2.2.15 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

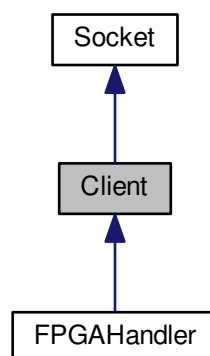
# Data Structure Documentation

### 5.1 Client Class Reference

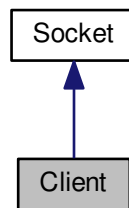
Base client object for the socket.

```
#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](mailto:laurent.forthomme@cern.ch)

## Date

24 Mar 2015

## 5.1.2 Constructor & Destructor Documentation

### 5.1.2.1 Client::Client ( ) [inline]

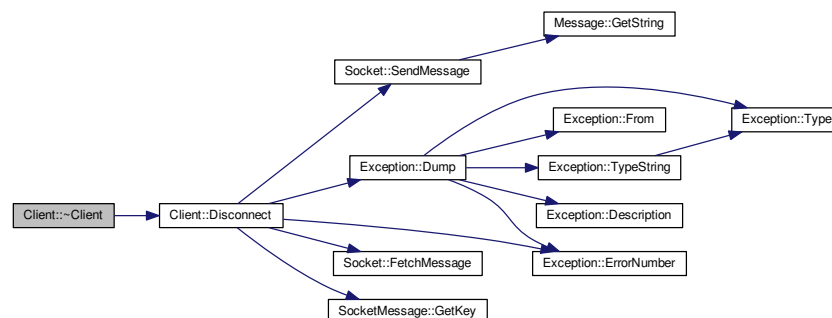
General void client constructor.

### 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:

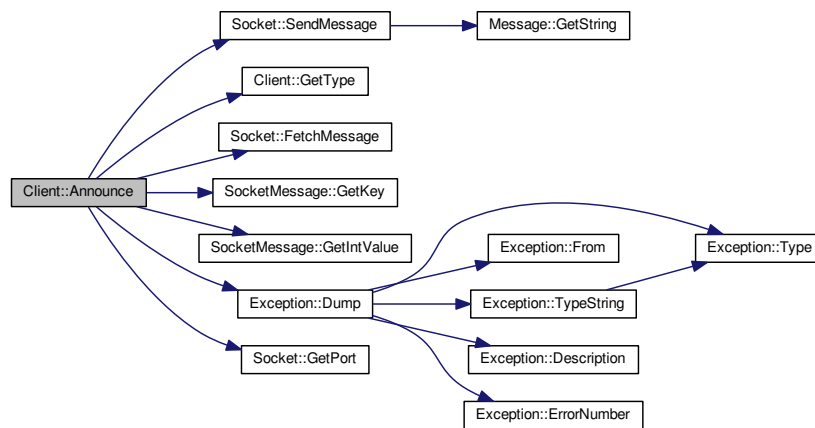


## 5.1.3 Member Function Documentation

### 5.1.3.1 void Client::Announce ( ) [private]

Announce our entry on the socket to its master.

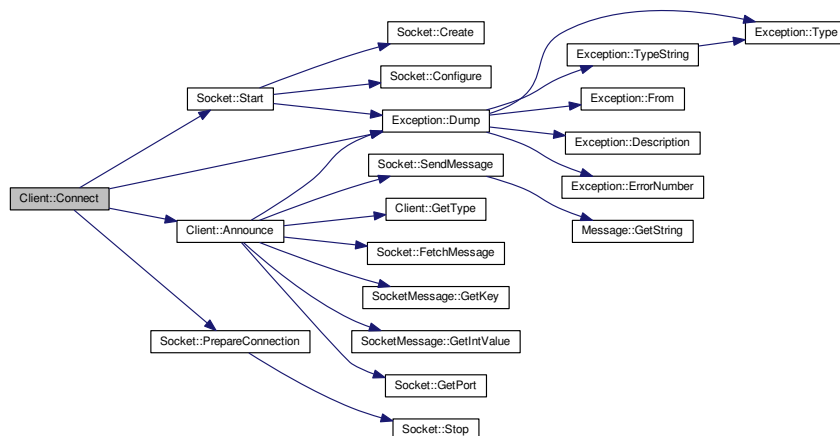
Here is the call graph for this function:



#### 5.1.3.2 bool Client::Connect ( )

Bind this client to the socket.

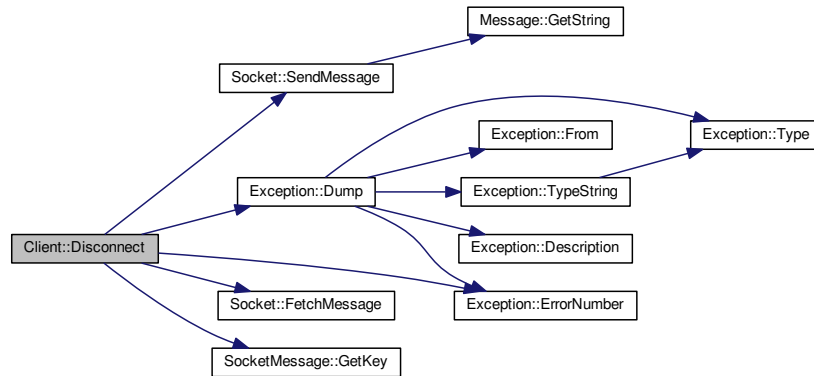
Here is the call graph for this function:



#### 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](#).

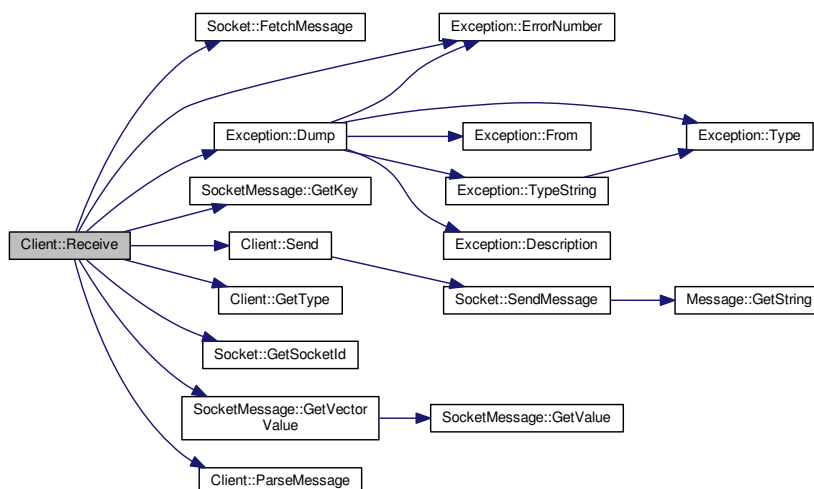
#### 5.1.3.5 virtual void Client::ParseMessage ( const SocketMessage & m ) [inline],[virtual]

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

#### 5.1.3.6 void Client::Receive ( )

Receive a socket message from the master.

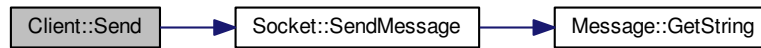
Here is the call graph for this function:



### 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](mailto: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:

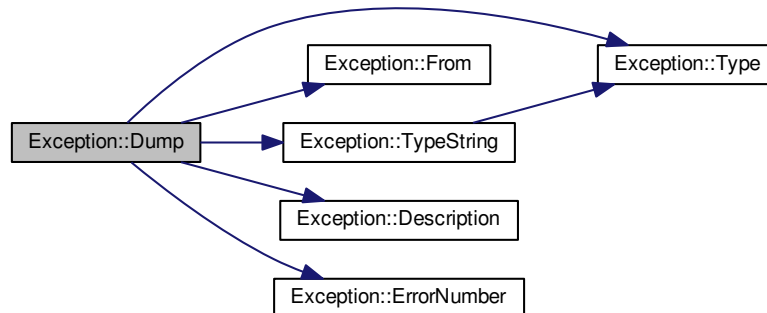


### 5.2.3 Member Function Documentation

**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:

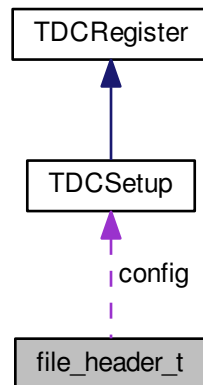
- `include/Exception.h`

## 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](mailto: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:

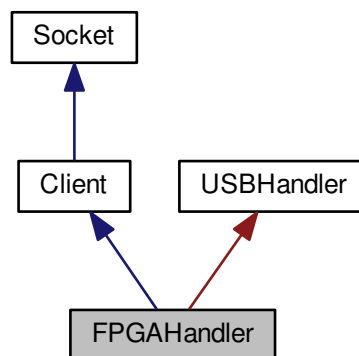
- include/FPGAHandler.h

## 5.4 FPGAHandler Class Reference

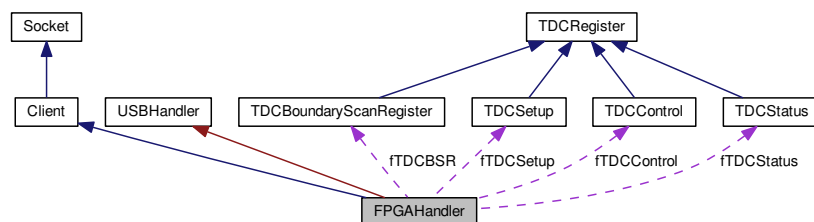
Driver for timing detectors' FPGA readout.

```
#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.*
  - bool [ErrorState](#) ()
  - 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 [SetRegister](#) (const [TDCControl::RegisterName](#) &r, unsigned int v)
  - unsigned int [ReadRegister](#) (const [TDCControl::RegisterName](#) &r)
  - void [ReadBoundaryScanRegister](#) ()

### Private Attributes

- std::string [fFilename](#)
- std::ofstream [fOutput](#)
- bool [flsFileOpen](#)
- bool [flsTDCInReadout](#)
- [TDCSetup](#) [fTDCSetup](#)
- [TDCControl](#) [fTDCControl](#)
- [TDCBoundaryScanRegister](#) [fTDCBSR](#)
- [TDCStatus](#) [fTDCStatus](#)

### 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](mailto:laurent.forthomme@cern.ch)

#### Date

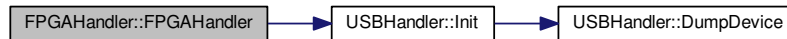
14 Apr 2015

## 5.4.2 Constructor & Destructor Documentation

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

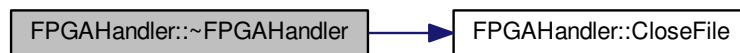
Bind to a FPGA through the USB protocol, and to the socket.

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 ( )

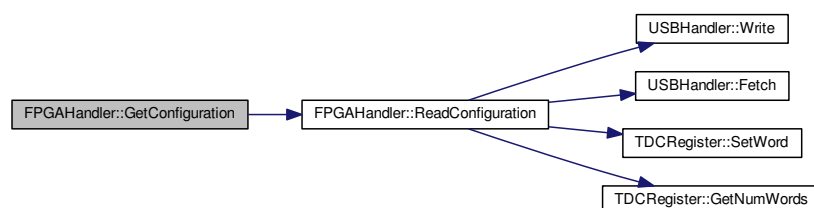
Close a previously opened output file used to store header/HPTDC events.

### 5.4.3.2 bool FPGAHandler::ErrorState ( )

### 5.4.3.3 TDCSetup FPGAHandler::GetConfiguration ( ) [inline]

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

Here is the call graph for this function:



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

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

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

[Socket](#) actor type retrieval method.

Reimplemented from [Client](#).

5.4.3.6 `void FPGAHandler::OpenFile ( )`

Open an output file to store header/HPTDC events.

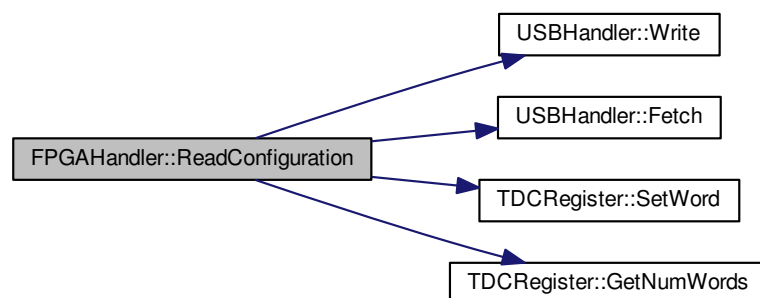
5.4.3.7 `void FPGAHandler::ReadBoundaryScanRegister ( )` `[private]`

5.4.3.8 `void FPGAHandler::ReadBuffer ( )`

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

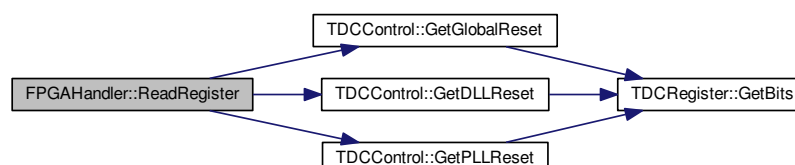
Read the setup word from the HPTDC internal setup register.

Here is the call graph for this function:



5.4.3.10 `unsigned int FPGAHandler::ReadRegister ( const TDCControl::RegisterName & r )` `[private]`

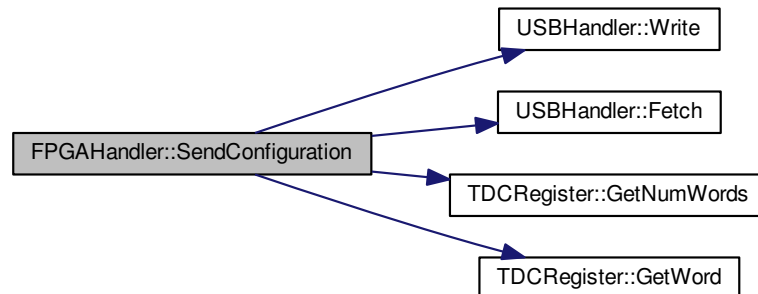
Here is the call graph for this function:



#### 5.4.3.11 void FPGAHandler::SendConfiguration ( ) [private]

Set the setup word to the HPTDC internal setup register.

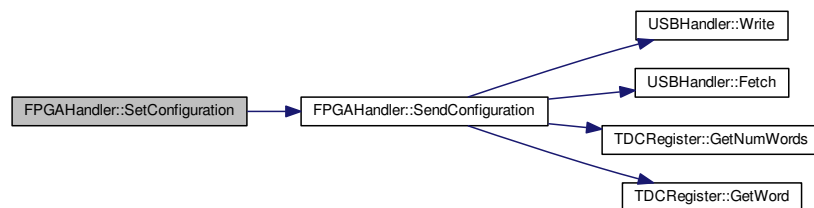
Here is the call graph for this function:



#### 5.4.3.12 void FPGAHandler::SetConfiguration ( const TDCSetup & c ) [inline]

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

Here is the call graph for this function:



#### 5.4.3.13 void FPGAHandler::SetRegister ( const TDCCControl::RegisterName & r, unsigned int v ) [private]

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 `TDCControl FPGAHandler::fTDCControl` [private]

5.4.4.7 `TDCSetup FPGAHandler::fTDCSetup` [private]

5.4.4.8 `TDCStatus FPGAHandler::fTDCStatus` [private]

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

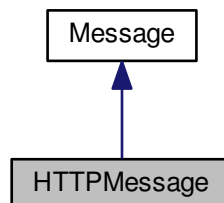
- `include/FPGAHandler.h`
- `src/FPGAHandler.cpp`

## 5.5 HTTPMessage Class Reference

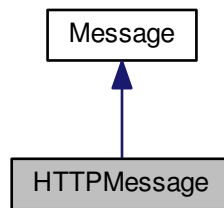
[Message](#) to be transmitted through a WebSocket protocol.

```
#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](mailto:laurent.forthomme@cern.ch)

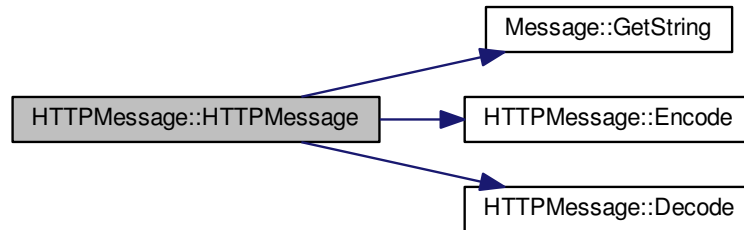
#### Date

1 Apr 2015

#### 5.5.2 Constructor & Destructor Documentation

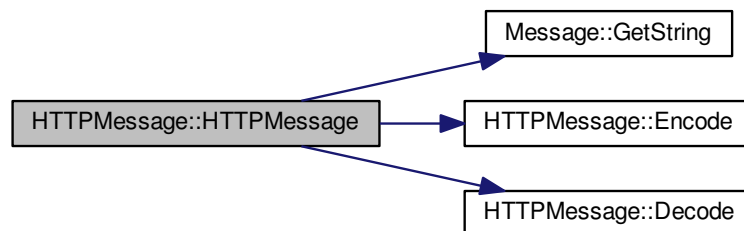
#### 5.5.2.1 HTTPMessage::HTTPMessage ( WebSocket \* *ws*, Message *m*, MessageAction *a* ) [inline]

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:

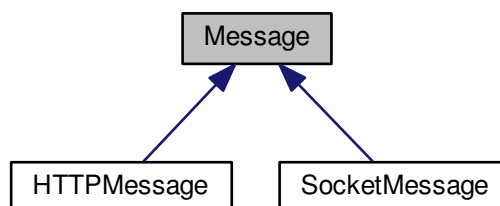
- `include/Messenger.h`

## 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](mailto: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:

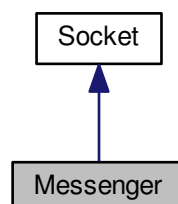
- include/Message.h

## 5.8 Messenger Class Reference

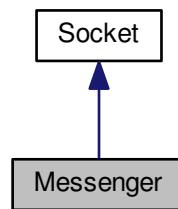
Base master object for the socket.

```
#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](mailto:laurent.forthomme@cern.ch)

#### Date

23 Mar 2015

### 5.8.2 Constructor & Destructor Documentation

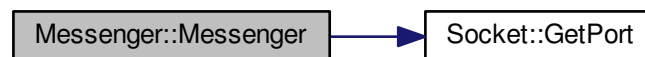
#### 5.8.2.1 Messenger::Messenger ( )

Build a void master object or socket actor.

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

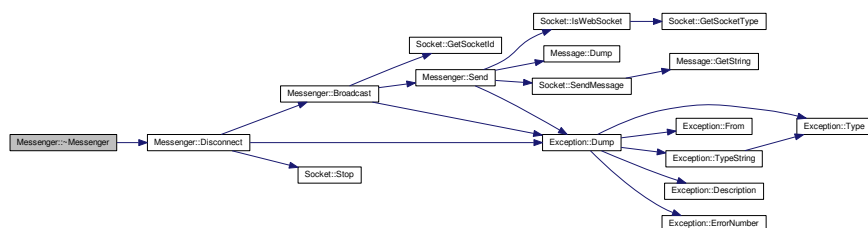
Build a master object to control the socket.

Here is the call graph for this function:



#### 5.8.2.3 Messenger::~~Messenger ( )

Here is the call graph for this function:



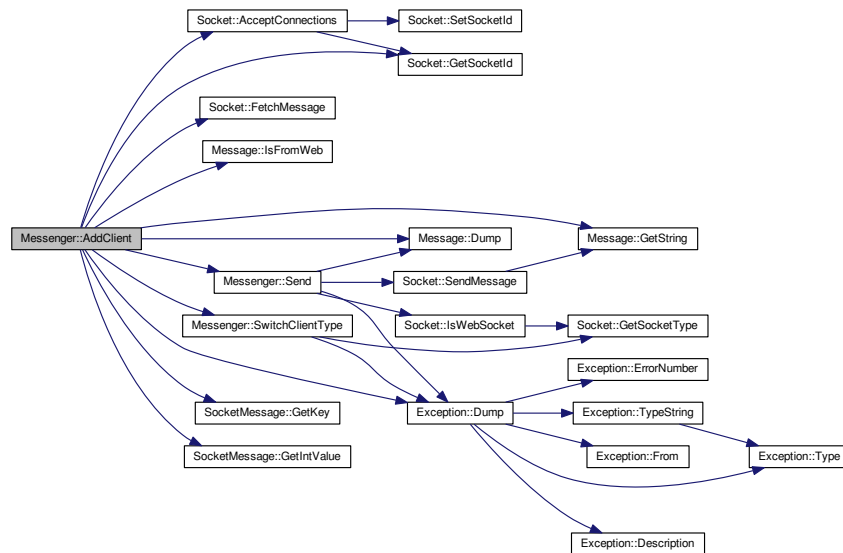
### 5.8.3 Member Function Documentation

#### 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:



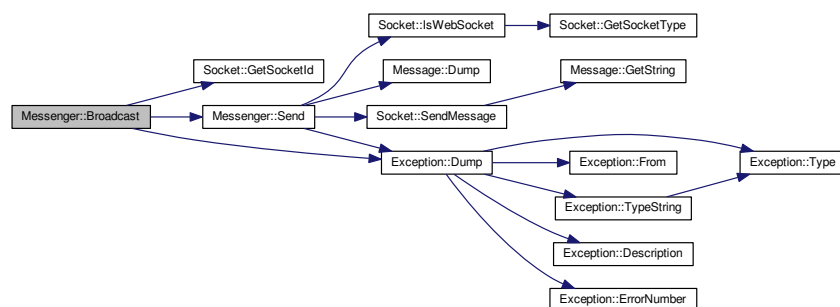
#### 5.8.3.2 void Messenger::Broadcast ( const Message & m ) const

Emit a message to all clients connected through the socket.

**Parameters**

in	<i>m</i>	<a href="#">Message</a> to transmit
----	----------	-------------------------------------

Here is the call graph for this function:

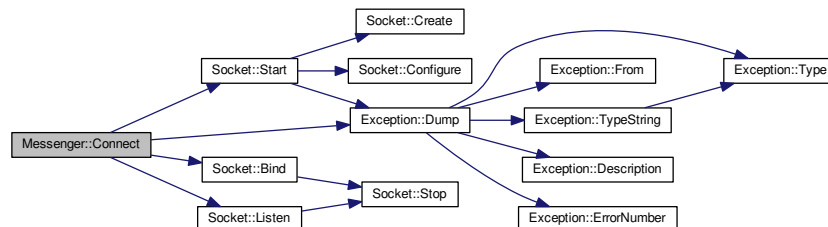


### 5.8.3.3 bool Messenger::Connect ( )

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:

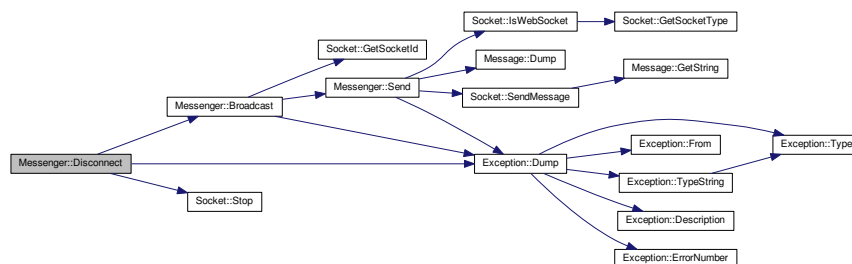


### 5.8.3.4 void Messenger::Disconnect ( )

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]

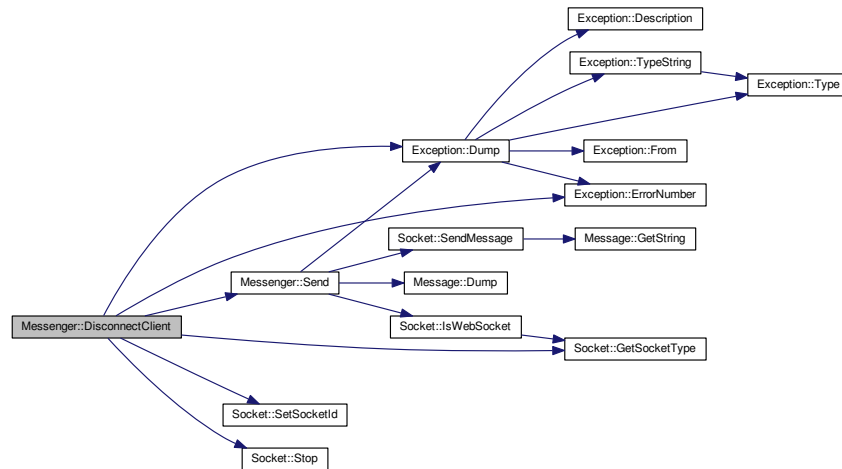
Disconnect a client.

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.

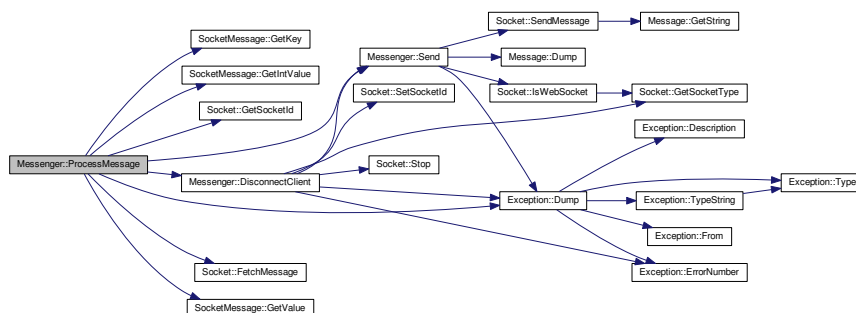
#### 5.8.3.7 void Messenger::ProcessMessage ( SocketMessage m, int sid ) [private]

Process a message received from the socket.

Parameters

in	Unique	identifier of the client sending the message
----	--------	--

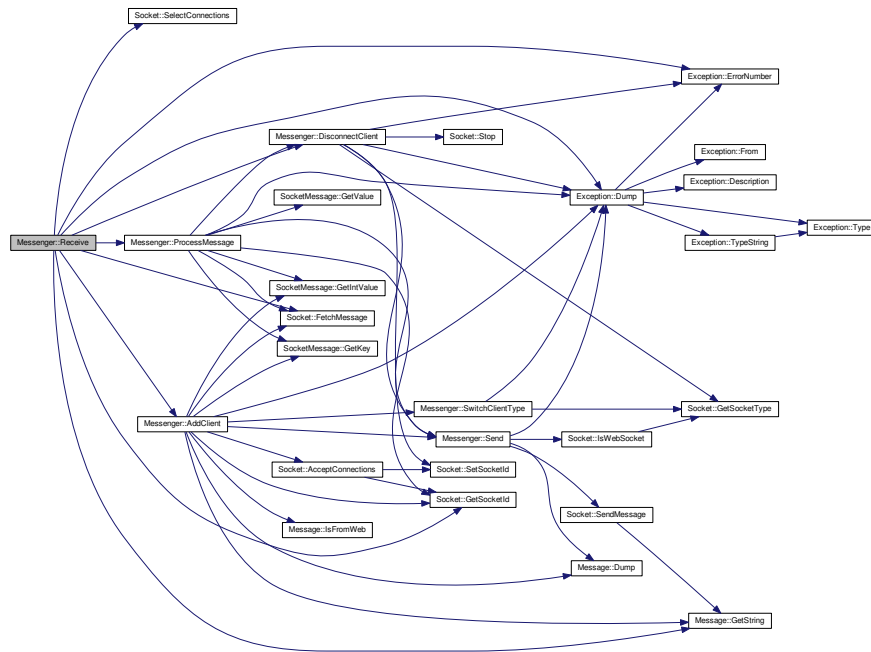
Here is the call graph for this function:



#### 5.8.3.8 void Messenger::Receive ( )

Handle a message reception from a client.

Here is the call graph for this function:



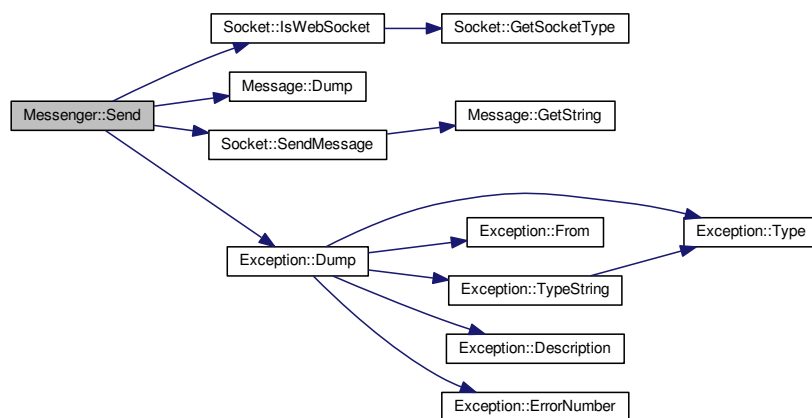
#### 5.8.3.9 void Messenger::Send ( const Message & m, int sid ) const [inline]

Send any type of message to any client.

##### Parameters

in	<i>m</i>	<a href="#">Message</a> to transmit
in	<i>sid</i>	Unique identifier of the client on this socket

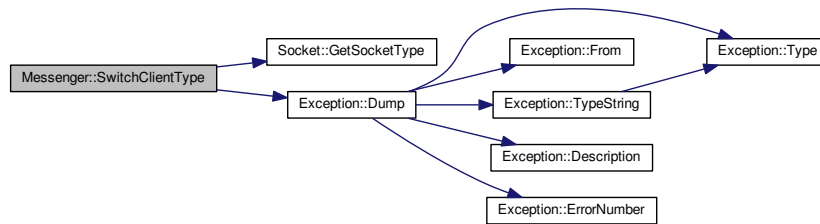
Here is the call graph for this function:





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:

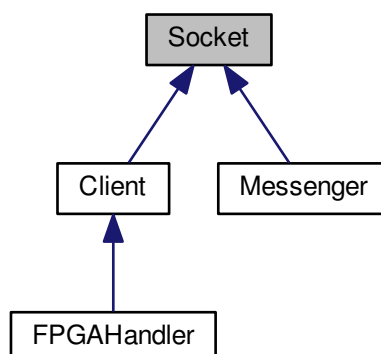
- include/Messenger.h
- src/Messenger.cpp

## 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](mailto: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 )`

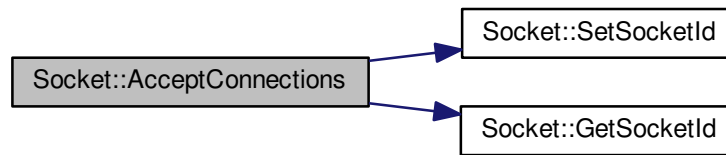
Accept connection from a client.

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.

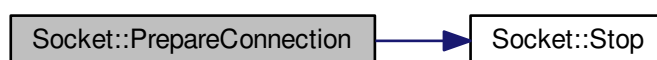
Set the socket to listen to any message coming from outside

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]

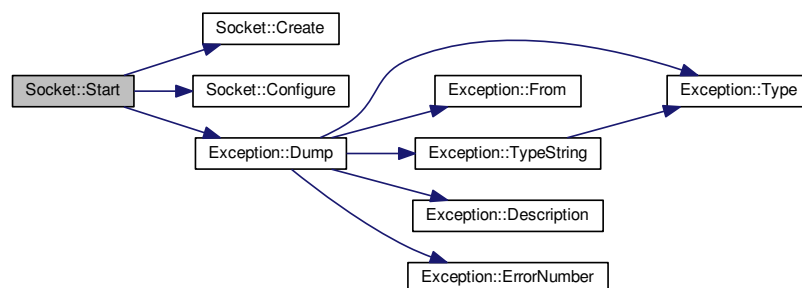
Start the socket.

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:

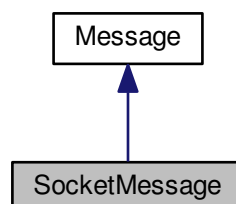
- include/Socket.h
- src/Socket.cpp

## 5.10 SocketMessage Class Reference

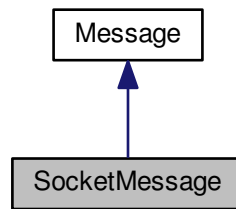
Socket-passed message type.

```
#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](mailto: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]`

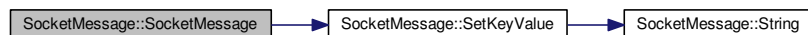
Here is the call graph for this function:



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

Construct a socket message out of a key.

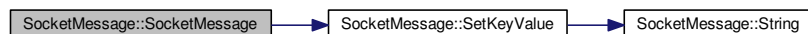
Here is the call graph for this function:



### 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.

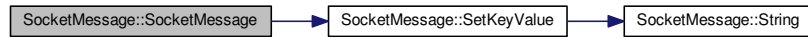
Here is the call graph for this function:



#### 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.

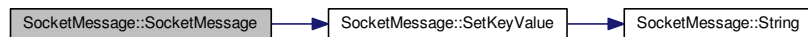
Here is the call graph for this function:



#### 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.

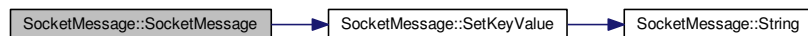
Here is the call graph for this function:



#### 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.

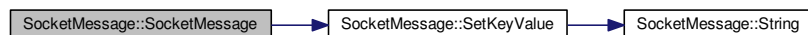
Here is the call graph for this function:



#### 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]

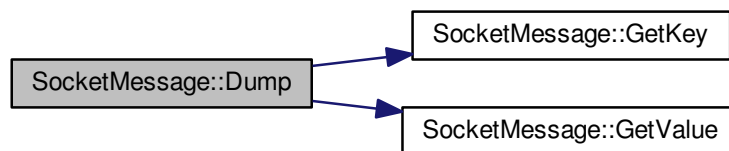
Construct a socket message out of a map of key/string-type value.

#### 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.

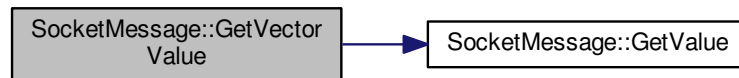
#### 5.10.3.5 std::string SocketMessage::GetValue ( ) const [inline]

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:

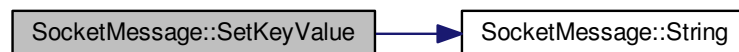


**5.10.3.7** `MessageMap SocketMessage::Object ( ) const` `[inline], [private]`

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

String-valued message.

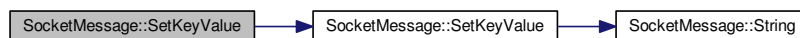
Here is the call graph for this function:



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

Send an integer-valued message.

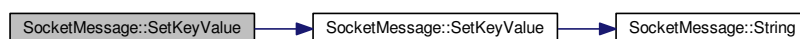
Here is the call graph for this function:



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

Float-valued message.

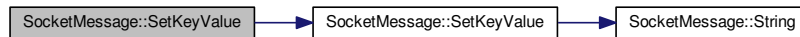
Here is the call graph for this function:



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]`

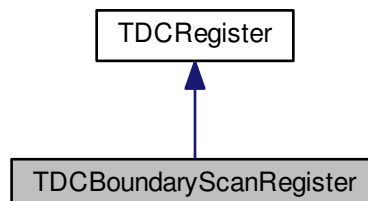
The documentation for this class was generated from the following file:

- include/SocketMessage.h

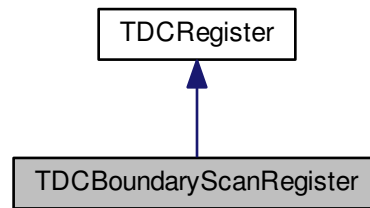
## 5.11 TDCBoundaryScanRegister Class Reference

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

Inheritance diagram for TDCBoundaryScanRegister:



Collaboration diagram for TDCBoundaryScanRegister:



### Public Member Functions

- [TDCBoundaryScanRegister](#) ()
- [TDCBoundaryScanRegister](#) (const [TDCBoundaryScanRegister](#) &bs)
- void [SetConstantValues](#) ()

### Additional Inherited Members

#### 5.11.1 Detailed Description

##### Author

Laurent Forthomme [laurent.forthomme@cern.ch](mailto:laurent.forthomme@cern.ch)

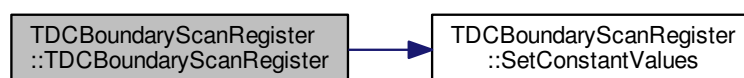
##### Date

24 Apr 2015

#### 5.11.2 Constructor & Destructor Documentation

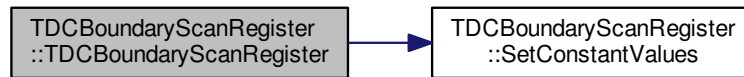
##### 5.11.2.1 TDCBoundaryScanRegister::TDCBoundaryScanRegister ( ) [inline]

Here is the call graph for this function:



### 5.11.2.2 TDCBoundaryScanRegister::TDCBoundaryScanRegister ( const TDCBoundaryScanRegister & bs ) [inline]

Here is the call graph for this function:



## 5.11.3 Member Function Documentation

### 5.11.3.1 void TDCBoundaryScanRegister::SetConstantValues ( ) [inline],[virtual]

Ensure that the critical constant values are properly set in the register word

Implements [TDCRegister](#).

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

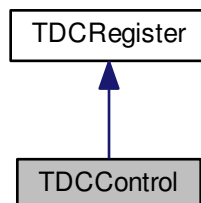
- include/TDCBoundaryScan.h

## 5.12 TDCControl Class Reference

Control word to be sent to the HPTDC chip.

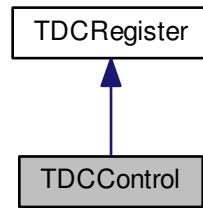
```
#include <TDCControl.h>
```

Inheritance diagram for TDCControl:





Collaboration diagram for TDCControl:



## Public Types

- enum [EnablePattern](#)
- enum [RegisterName](#) { [R\\_EnablePattern](#), [R\\_GlobalReset](#), [R\\_DLLReset](#), [R\\_PLLReset](#) }

## Public Member Functions

- [TDCControl](#) ()
- [TDCControl](#) (const [TDCControl](#) &c)
- void [SetEnablePattern](#) (const [EnablePattern](#) &ep)
- [EnablePattern](#) [GetEnablePattern](#) () const
- void [SetGlobalReset](#) (const bool gr=true)
- bool [GetGlobalReset](#) () const
- void [SetDLLReset](#) (const bool dr=true)
- bool [GetDLLReset](#) () const
- void [SetPLLReset](#) (const bool pr=true)
- bool [GetPLLReset](#) () const
- void [EnableChannel](#) (unsigned int id)
- void [EnableAllChannels](#) ()
- void [DisableChannel](#) (unsigned int id)
- void [DisableAllChannels](#) ()
- void [Dump](#) (int verb=1, std::ostream &os=std::cout) const
- void [SetConstantValues](#) ()

## Private Member Functions

- void [SetControlParity](#) (const bool cp=true)

## 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 Detailed Description

Control word to be sent to the HPTDC chip.

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

#### Author

Laurent Forthomme [laurent.forthomme@cern.ch](mailto:laurent.forthomme@cern.ch)

#### Date

24 Apr 2015

### 5.12.2 Constructor & Destructor Documentation

#### 5.12.2.1 TDCCControl::TDCCControl ( ) [inline]

Here is the call graph for this function:



#### 5.12.2.2 TDCCControl::TDCCControl ( const TDCCControl & c ) [inline]

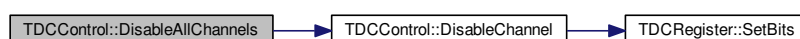
Here is the call graph for this function:



### 5.12.3 Member Function Documentation

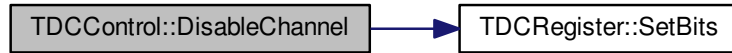
#### 5.12.3.1 void TDCCControl::DisableAllChannels ( ) [inline]

Here is the call graph for this function:



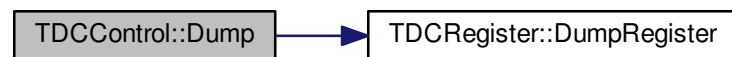
### 5.12.3.2 void TDCCControl::DisableChannel ( unsigned int *id* ) [inline]

Here is the call graph for this function:



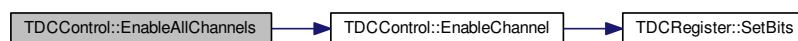
### 5.12.3.3 void TDCCControl::Dump ( int *verb* = 1, std::ostream & *os* = std::cout ) const [inline]

Here is the call graph for this function:



### 5.12.3.4 void TDCCControl::EnableAllChannels ( ) [inline]

Here is the call graph for this function:



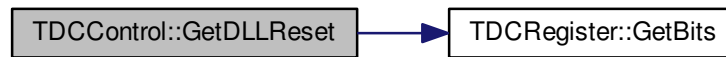
### 5.12.3.5 void TDCCControl::EnableChannel ( unsigned int *id* ) [inline]

Here is the call graph for this function:



#### 5.12.3.6 `bool TDCControl::GetDLLReset ( ) const [inline]`

Here is the call graph for this function:



#### 5.12.3.7 `EnablePattern TDCControl::GetEnablePattern ( ) const [inline]`

Here is the call graph for this function:



#### 5.12.3.8 `bool TDCControl::GetGlobalReset ( ) const [inline]`

Here is the call graph for this function:



#### 5.12.3.9 `bool TDCControl::GetPLLReset ( ) const [inline]`

Here is the call graph for this function:



#### 5.12.3.10 `void TDCControl::SetConstantValues ( ) [inline],[virtual]`

Ensure that the critical constant values are properly set in the register word

Implements [TDCRegister](#).

Here is the call graph for this function:



#### 5.12.3.11 `void TDCControl::SetControlParity ( const bool cp = true ) [inline],[private]`

Here is the call graph for this function:



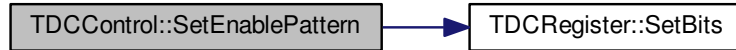
#### 5.12.3.12 `void TDCControl::SetDLLReset ( const bool dr = true ) [inline]`

Here is the call graph for this function:



5.12.3.13 `void TDCControl::SetEnablePattern ( const EnablePattern & ep ) [inline]`

Here is the call graph for this function:



5.12.3.14 `void TDCControl::SetGlobalReset ( const bool gr=true ) [inline]`

Here is the call graph for this function:



5.12.3.15 `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 TDCCControl::kPLLReset = 38` [static], [private]

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

- include/TDCCControl.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](mailto: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:



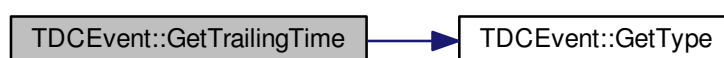
#### 5.13.3.5 `unsigned int TDCEvent::GetTDCId ( ) const [inline]`

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]

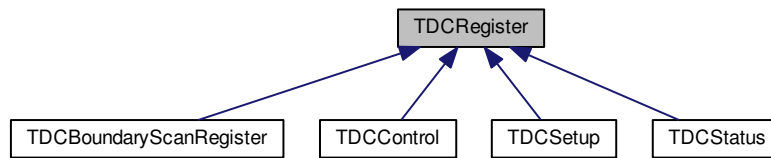
The documentation for this class was generated from the following file:

- include/TDCEvent.h

## 5.14 TDCRegister Class Reference

```
#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 unsigned int size)
- [TDCRegister](#) (const unsigned int size, const [TDCRegister](#) &r)
- 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](#) (std::ostream &os=std::cout, const [bit](#) max\_bits=-1) const
- virtual void [SetConstantValues](#) ()=0

## 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.*
- void [Clear](#) ()  
*Set all bits in this register to '0'.*

## Protected Attributes

- [word\\_t](#) \* [fWord](#)  
*Pointer to this register's word.*
- unsigned int [fNumWords](#)
- unsigned int [fWordSize](#)  
*Number of bits in this register.*

### 5.14.1 Member Typedef Documentation

#### 5.14.1.1 `typedef uint16_t TDCRegister::bit`

LSB index.

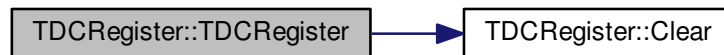
#### 5.14.1.2 `typedef uint32_t TDCRegister::word_t`

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

### 5.14.2 Constructor & Destructor Documentation

#### 5.14.2.1 `TDCRegister::TDCRegister ( const unsigned int size ) [inline]`

Here is the call graph for this function:



#### 5.14.2.2 `TDCRegister::TDCRegister ( const unsigned int size, const TDCRegister & r ) [inline]`

Here is the call graph for this function:



#### 5.14.2.3 `virtual TDCRegister::~~TDCRegister ( ) [inline], [virtual]`

### 5.14.3 Member Function Documentation

#### 5.14.3.1 `void TDCRegister::Clear ( ) [inline], [protected]`

Set all bits in this register to '0'.

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

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

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

Number of words in the register.

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

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

Retrieve one subset from the register word.

5.14.3.6 `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

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

5.14.3.7 `virtual void TDCRegister::SetConstantValues ( )` `[pure virtual]`

Ensure that the critical constant values are properly set in the register word

Implemented in [TDCSetup](#), [TDCControl](#), [TDCBoundaryScanRegister](#), and [TDCStatus](#).

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

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

## 5.14.4 Field Documentation

5.14.4.1 `unsigned int TDCRegister::fNumWords` `[protected]`

Number of words to fit the *fWordSize* bits of this register to this object

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

Pointer to this register's word.

#### 5.14.4.3 unsigned int TDCRegister::fWordSize [protected]

Number of bits in this register.

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

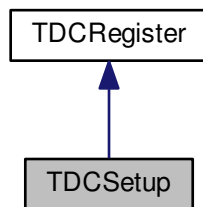
- include/TDCRegister.h

## 5.15 TDCSetup Class Reference

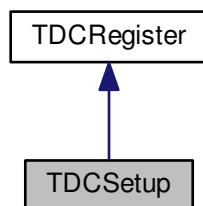
Setup word to be sent to the HPTDC chip.

```
#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)
- void SetEnableErrorMark (const bool em)  
*Mark events with error if global error signal is set.*
- bool GetEnableErrorMark () const
- void SetEnableErrorBypass (const bool eb)  
*Bypass TDC chip if global error signal is set.*
- bool GetEnableErrorBypass () const
- void SetEnableError (const uint16\_t &err)  
*Enable internal error types for generation of global error signals.*
- uint16\_t GetEnableError () const
- void SetEnableSerial (const bool es)  
*Enable of serial read-out (otherwise parallel read-out)*
- bool GetEnableSerial () const
- void SetEnableJTAGReadout (const bool jr)  
*Enable of read-out via JTAG.*
- bool GetEnableJTAGReadout () const
- void SetReadoutFIFOSize (int rfs)  
*Effective size of readout FIFO.*
- int GetReadoutFIFOSize () const
- void SetRejectCountOffset (uint16\_t rco)  
*Set the offset in reject counter (defines reject latency together with coarse count offset)*
- uint16\_t GetRejectCountOffset () const  
*Extract the offset in reject counter.*
- void SetSearchWindow (uint16\_t sw)  
*Set the search window (in multiples of clock cycles: 0=25 ns, 1=50 ns, ...)*

- `uint16_t GetSearchWindow () const`  
*Extract the search window (in multiples of clock cycles: 0=25 ns, 1=50 ns, ...)*
- `void SetMatchWindow (uint16_t mw)`  
*Set the matching window (in multiples of clock cycles: 0=25 ns, 1=50 ns, ...)*
- `uint16_t GetMatchWindow () const`  
*Extract the matching window (in multiples of clock cycles: 0=25 ns, 1=50 ns, ...)*
- `void SetEdgeResolution (const EdgeResolution r)`
- `EdgeResolution GetEdgeResolution () const`
- `void SetMaxEventSize (int sz=-1)`  
*Set the maximum number of hits per event.*
- `uint8_t GetMaxEventSize () const`  
*Extract the maximum number of hits per event.*
- `void SetRejectFIFOFull (const bool rej=true)`  
*Reject hits when readout FIFO full.*
- `bool GetRejectFIFOFull () const`  
*Are hits rejected when readout FIFO is full?*
- `void SetEnableReadoutOccupancy (const bool ro=true)`  
*Enable the readout of buffer occupancies for each event (for debugging purposes)*
- `bool GetEnableReadoutOccupancy () const`
- `void SetEnableReadoutSeparator (const bool ro=true)`  
*Enable the readout of separators for each event (for debugging purposes, valid if readout of occupancies is enabled)*
- `bool GetEnableReadoutSeparator () const`
- `void SetEventCountOffset (uint16_t eco)`  
*Set offset for the event counter.*
- `void SetTriggerCountOffset (uint16_t tco)`  
*Set offset for the trigger time tag counter to set effective trigger latency.*
- `uint16_t GetTriggerCountOffset () const`  
*Extract trigger time tag count offset.*
- `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  $\sim 10$  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 setup word provided by/to the HPTDC chip

#### Author

Laurent Forthomme [laurent.forthomme@cern.ch](mailto:laurent.forthomme@cern.ch)

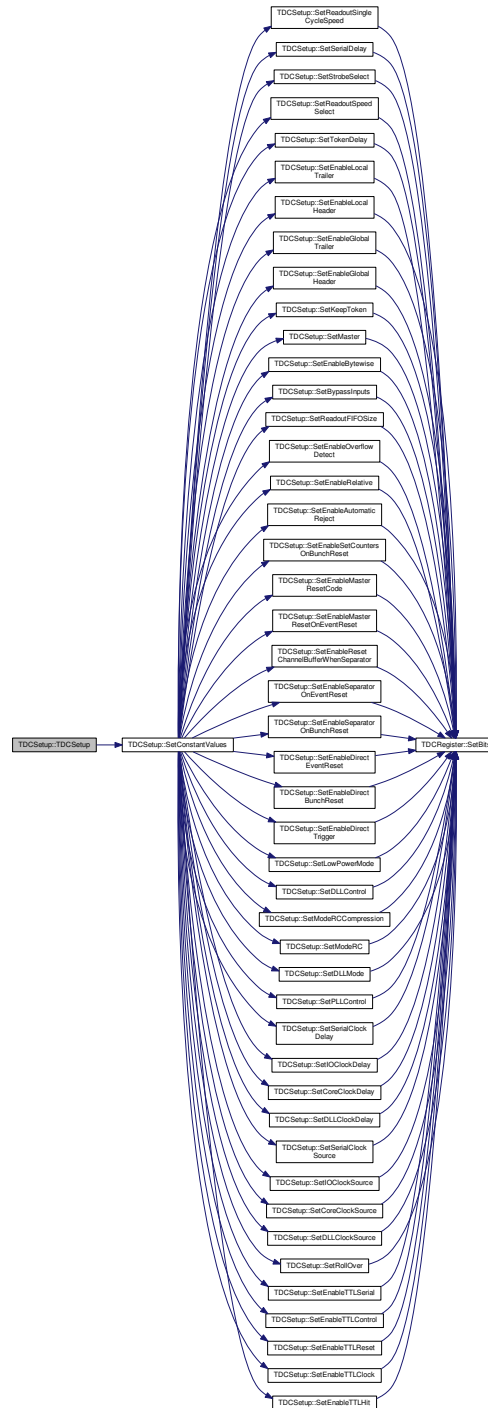
#### Date

16 Apr 2015

### 5.15.2 Constructor & Destructor Documentation

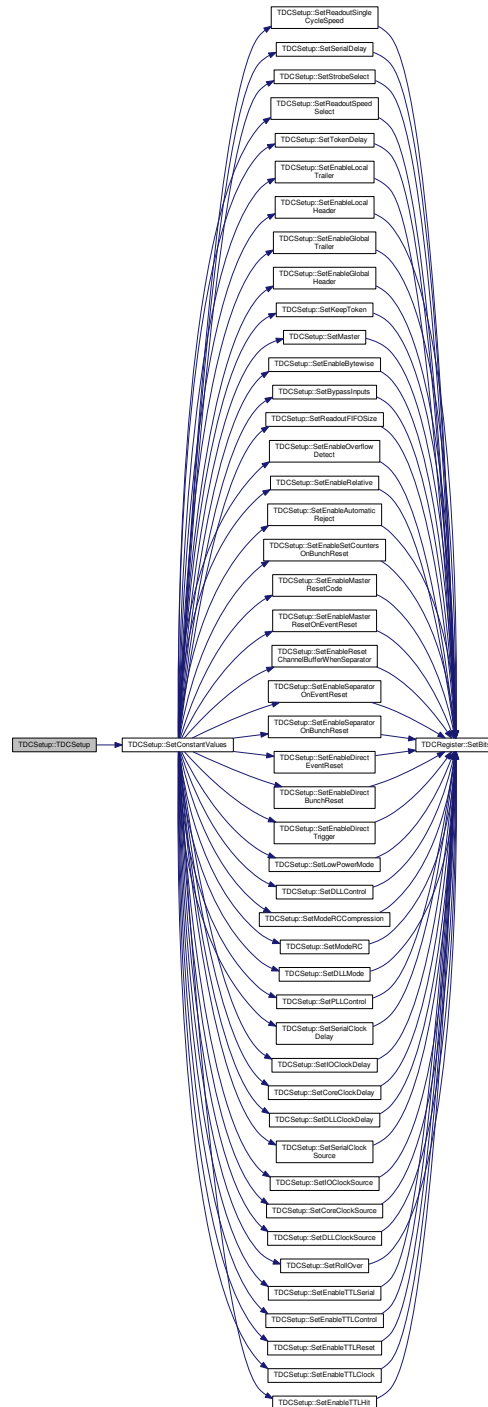
### 5.15.2.1 TDCSetup::TDCSetup ( ) [inline]

Here is the call graph for this function:



## 5.15.2.2 TDCSetup::TDCSetup ( const TDCSetup &amp; c ) [inline]

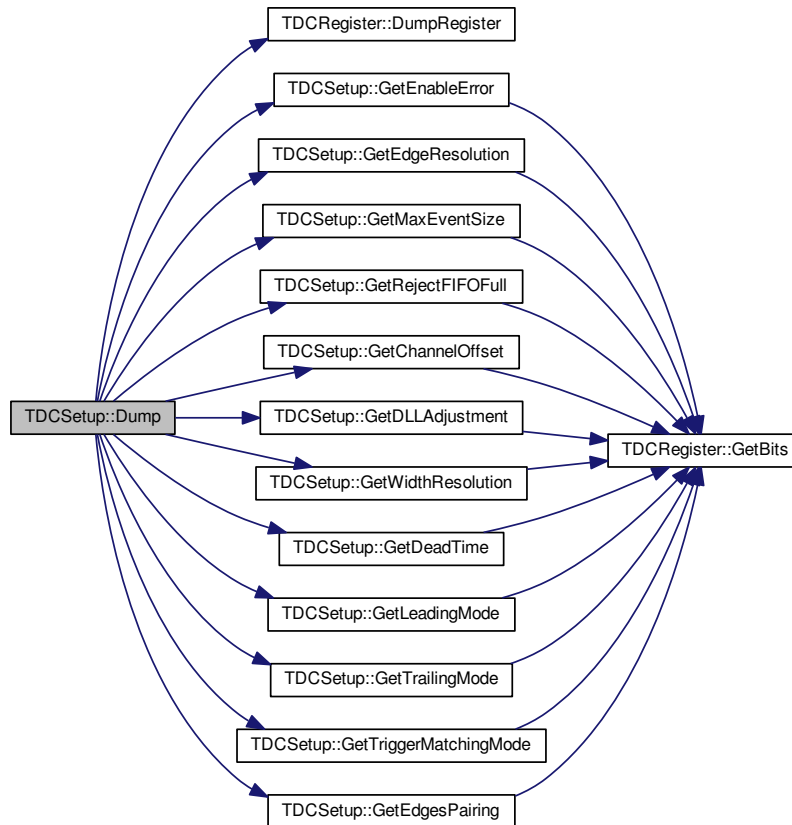
Here is the call graph for this function:



## 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:

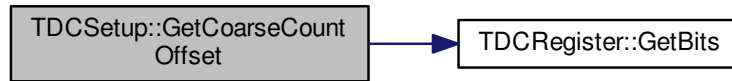


### 5.15.3.3 uint16\_t TDCSetup::GetCoarseCountOffset ( ) const [inline]

Extract offset for the coarse time counter.



Here is the call graph for this function:



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

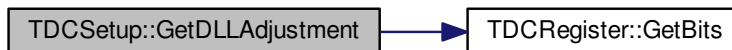
Here is the call graph for this function:



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

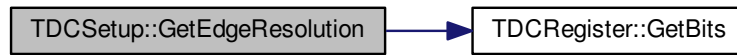
Set the adjustment of DLL taps.

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:



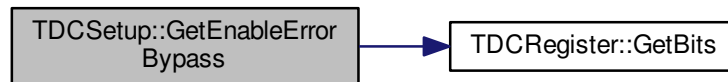
#### 5.15.3.8 **uint16\_t** TDCSetup::GetEnableError ( ) const [inline]

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:



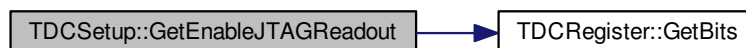
#### 5.15.3.10 bool TDCSetup::GetEnableErrorMark ( ) const [inline]

Here is the call graph for this function:



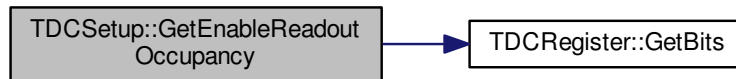
#### 5.15.3.11 bool TDCSetup::GetEnableJTAGReadout ( ) const [inline]

Here is the call graph for this function:



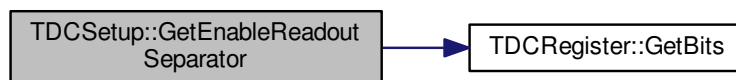
#### 5.15.3.12 `bool TDCSetup::GetEnableReadoutOccupancy ( ) const [inline]`

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]`

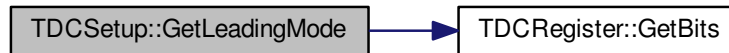
Here is the call graph for this function:



#### 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, ...)

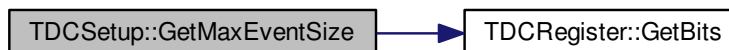
Here is the call graph for this function:



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

Extract the maximum number of hits per event.

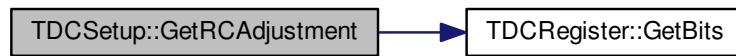
Here is the call graph for this function:



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

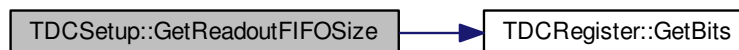
Extract the adjustment of the RC delay line.

Here is the call graph for this function:



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

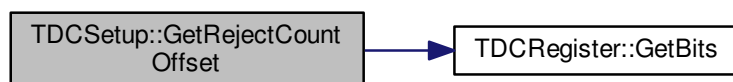
Here is the call graph for this function:



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.

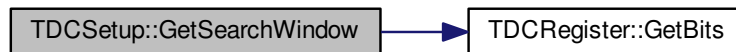
Here is the call graph for this function:



#### 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:

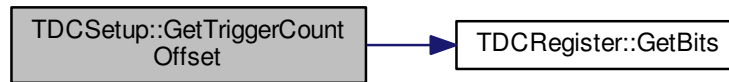


#### 5.15.3.27 `uint16_t TDCSetup::GetTriggerCountOffset ( ) const [inline]`

Extract trigger time tag count offset.



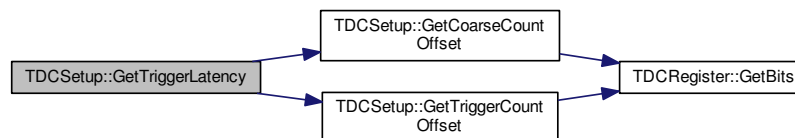
Here is the call graph for this function:



#### 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)

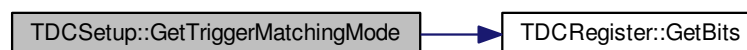
Here is the call graph for this function:



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

Extract the enable status of trigger matching mode.

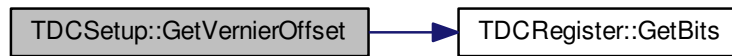
Here is the call graph for this function:



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

Extract the offset in vernier decoding.

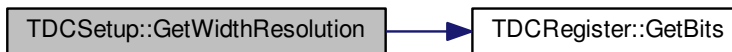
Here is the call graph for this function:



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

Extract the pulse width resolution when paired measurements are performed.

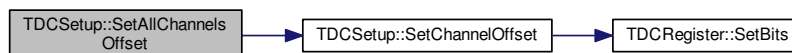
Here is the call graph for this function:



#### 5.15.3.32 void TDCSetup::SetAllChannelsOffset ( uint16\_t offset ) [inline]

Set the time offset for all channels.

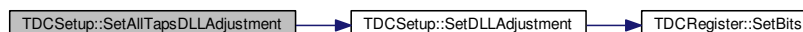
Here is the call graph for this function:



#### 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.

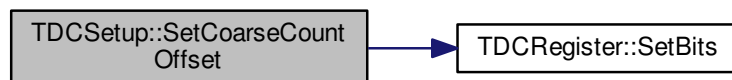
Here is the call graph for this function:



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:

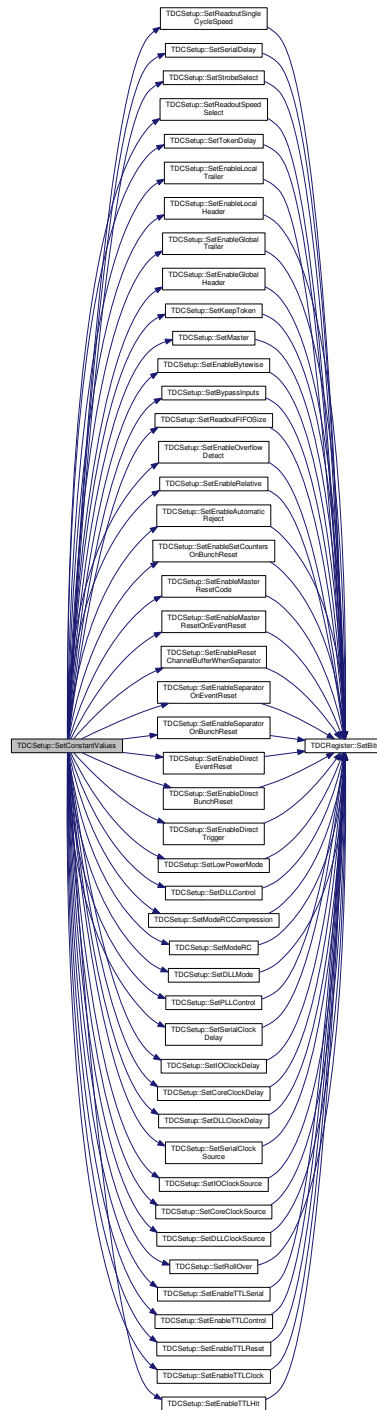


5.15.3.37 `void TDCSetup::SetConstantValues ( ) [virtual]`

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

Implements [TDCRegister](#).

Here is the call graph for this function:



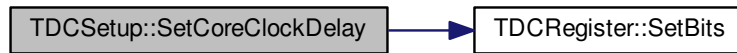
**5.15.3.38** void TDCSetup::SetCoreClockDelay ( const bool *delay\_clock*, const uint8\_t *delay* ) [inline], [private]

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.

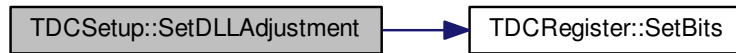
Here is the call graph for this function:



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

Set the DLL taps adjustments with a resolution of  $\sim 10$  ps.

Here is the call graph for this function:



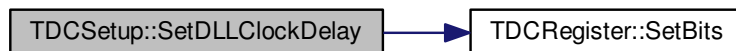
**5.15.3.42** `void TDCSetup::SetDLLClockDelay ( const bool delay_clock, const uint8_t delay )` `[inline],[private]`

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

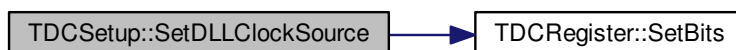
Here is the call graph for this function:



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

Selection of clock source for DLL.

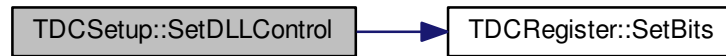
Here is the call graph for this function:



**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)

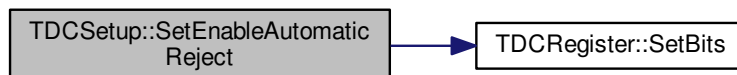
Here is the call graph for this function:



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

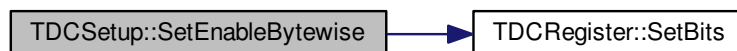
Enable of automatic rejection (should always be enabled if trigger matching mode!)

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:

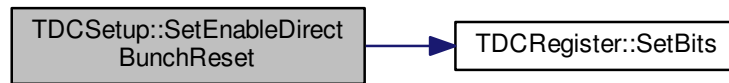


5.15.3.50 `void TDCSetup::SetEnableDirectBunchReset ( const bool edbr = true ) [inline],[private]`

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



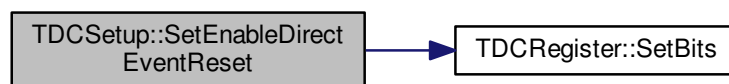
Here is the call graph for this function:



**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.

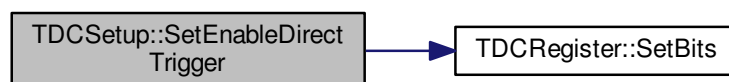
Here is the call graph for this function:



**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.

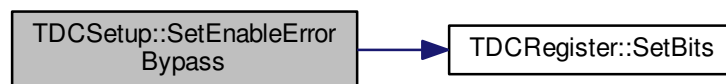
Here is the call graph for this function:



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.

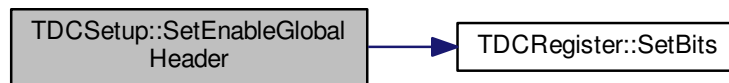
Here is the call graph for this function:



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)

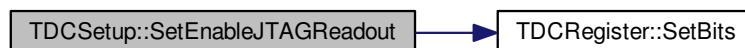
Here is the call graph for this function:



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

Enable of read-out via JTAG.

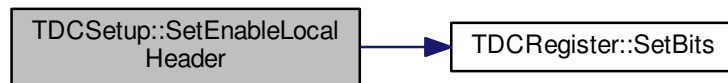
Here is the call graph for this function:



**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.

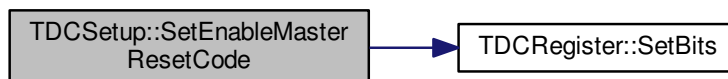
Here is the call graph for this function:



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

Enable master reset code on encoded\_control.

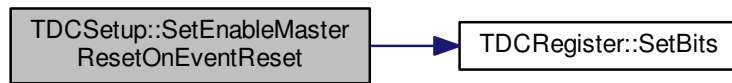
Here is the call graph for this function:



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

Enable master reset of whole TDC on event reset.

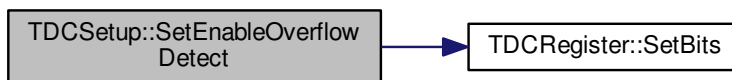
Here is the call graph for this function:



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

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

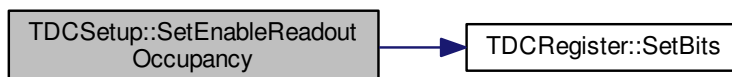
Here is the call graph for this function:



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

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

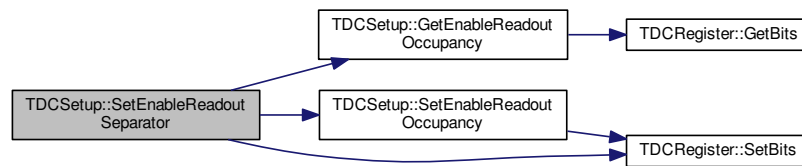
Here is the call graph for this function:



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)

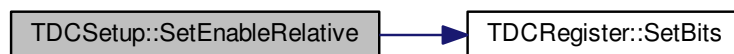
Here is the call graph for this function:



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.

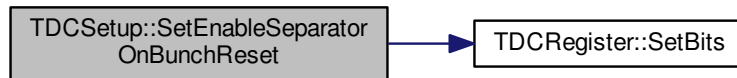
Here is the call graph for this function:



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

Enable generation of separator on bunch reset.

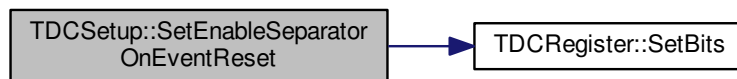
Here is the call graph for this function:



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)

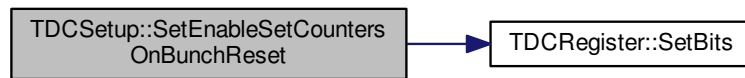
Here is the call graph for this function:



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

Enable all counters to be set on bunch count reset.

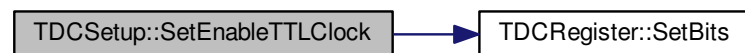
Here is the call graph for this function:



**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.

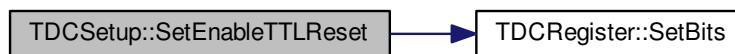
Here is the call graph for this function:



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:



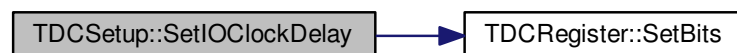
**5.15.3.78** void TDCSetup::SetIOClockDelay ( const bool *delay\_clock*, const uint8\_t *delay* ) [inline],[private]

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

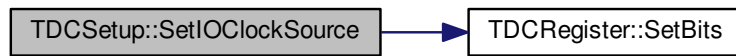
Here is the call graph for this function:



**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.

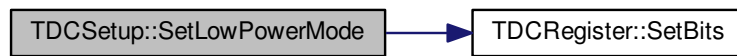
Here is the call graph for this function:



#### 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:

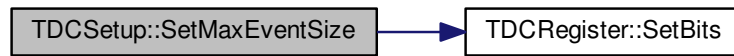


5.15.3.85 `void TDCSetup::SetMaxEventSize ( int sz=-1 ) [inline]`

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.

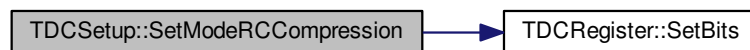
Here is the call graph for this function:



**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.

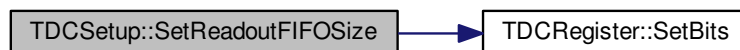
Here is the call graph for this function:



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

Effective size of readout FIFO.

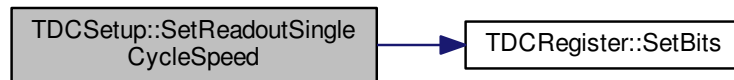
Here is the call graph for this function:



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

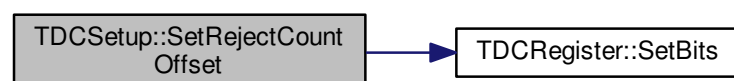
Here is the call graph for this function:



**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:

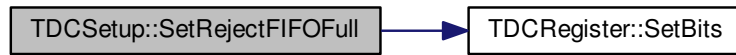


5.15.3.94 `void TDCSetup::SetRejectFIFOFull ( const bool rej = true ) [inline]`

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]`

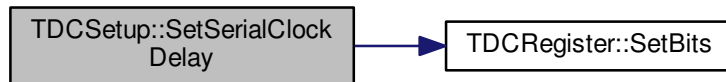
Delay of internal serial clock.



## 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

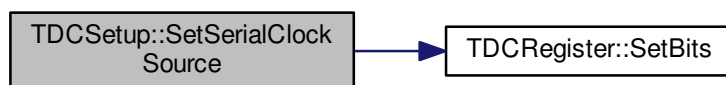
Here is the call graph for this function:



**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  $\sim 1$  ns.

Here is the call graph for this function:



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

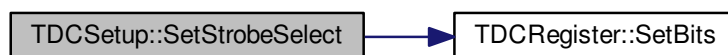
Set the parity of setup data (should be an even parity)

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  $\sim 1$  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.

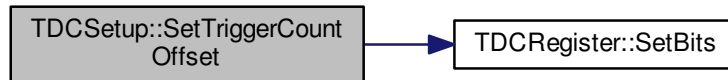
Here is the call graph for this function:



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

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

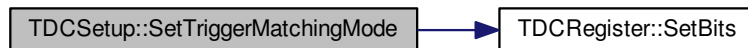
Here is the call graph for this function:



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.

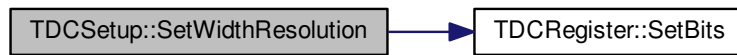
Here is the call graph for this function:



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]`

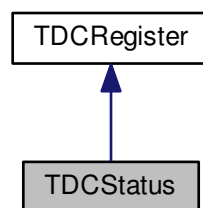
The documentation for this class was generated from the following files:

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

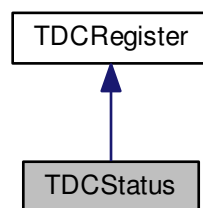
## 5.16 TDCStatus Class Reference

```
#include <TDCStatus.h>
```

Inheritance diagram for TDCStatus:



Collaboration diagram for TDCStatus:



### Public Member Functions

- [TDCStatus](#) ()
- [TDCStatus](#) (const [TDCStatus](#) &s)
- void [SetConstantValues](#) ()

### Static Private Attributes

- static const [bit](#) [kError](#) = 0
- static const [bit](#) [kHaveToken](#) = 11



- static const [bit kReadoutFIFOOccupancy](#) = 12
- static const [bit kReadoutFIFOFull](#) = 20
- static const [bit kReadoutFIFOEmpty](#) = 21
- static const [bit kL1Occupancy](#) = 22
- static const [bit kTriggerFIFOOccupancy](#) = 54
- static const [bit kTriggerFIFOFull](#) = 58
- static const [bit kTriggerFIFOEmpty](#) = 59
- static const [bit kDLLLock](#) = 60

## Additional Inherited Members

### 5.16.1 Detailed Description

#### Author

Laurent Forthomme [laurent.forthomme@cern.ch](mailto:laurent.forthomme@cern.ch)

#### Date

27 Apr 2015

### 5.16.2 Constructor & Destructor Documentation

#### 5.16.2.1 TDCStatus::TDCStatus ( ) [inline]

Here is the call graph for this function:



#### 5.16.2.2 TDCStatus::TDCStatus ( const TDCStatus & s ) [inline]

Here is the call graph for this function:



### 5.16.3 Member Function Documentation

5.16.3.1 `void TDCStatus::SetConstantValues ( ) [inline],[virtual]`

Ensure that the critical constant values are properly set in the register word

Implements [TDCRegister](#).

## 5.16.4 Field Documentation

5.16.4.1 `const bit TDCStatus::kDLLLock = 60 [static],[private]`

5.16.4.2 `const bit TDCStatus::kError = 0 [static],[private]`

5.16.4.3 `const bit TDCStatus::kHaveToken = 11 [static],[private]`

5.16.4.4 `const bit TDCStatus::kL1Occupancy = 22 [static],[private]`

5.16.4.5 `const bit TDCStatus::kReadoutFIFOEmpty = 21 [static],[private]`

5.16.4.6 `const bit TDCStatus::kReadoutFIFOFull = 20 [static],[private]`

5.16.4.7 `const bit TDCStatus::kReadoutFIFOOccupancy = 12 [static],[private]`

5.16.4.8 `const bit TDCStatus::kTriggerFIFOEmpty = 59 [static],[private]`

5.16.4.9 `const bit TDCStatus::kTriggerFIFOFull = 58 [static],[private]`

5.16.4.10 `const bit TDCStatus::kTriggerFIFOOccupancy = 54 [static],[private]`

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

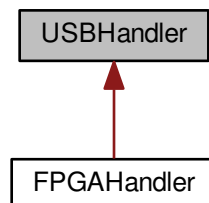
- `include/TDCStatus.h`

## 5.17 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](#)

### 5.17.1 Detailed Description

Generic USB communication handler.

#### Date

21 Apr 2015

#### Author

Laurent Forthomme [laurent.forthomme@cern.ch](mailto:laurent.forthomme@cern.ch)

### 5.17.2 Constructor & Destructor Documentation

5.17.2.1 [USBHandler::USBHandler](#) ( const char \* dev )

5.17.2.2 [virtual USBHandler::~~USBHandler](#) ( ) [\[inline\]](#), [\[virtual\]](#)

### 5.17.3 Member Function Documentation

5.17.3.1 [void USBHandler::DumpDevice](#) ( libusb\_device \* dev, int verb = 1, std::ostream & out = std::cout )

5.17.3.2 [uint32\\_t USBHandler::Fetch](#) ( uint8\_t size ) const [\[inline\]](#), [\[protected\]](#)

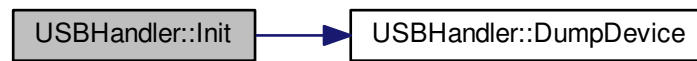
Receive a word from the USB device.

5.17.3.3 [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:



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

Write a word to the USB device.

#### 5.17.4 Field Documentation

5.17.4.1 `std::string USBHandler::fDevice` `[private]`

5.17.4.2 `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, [17](#)
- ~Exception
  - Exception, [21](#)
- ~FPGAHandler
  - FPGAHandler, [26](#)
- ~Message
  - Message, [33](#)
- ~Messenger
  - Messenger, [36](#)
- ~Socket
  - Socket, [43](#)
- ~SocketMessage
  - SocketMessage, [52](#)
- ~TDCEvent
  - TDCEvent, [64](#)
- ~TDCRegister
  - TDCRegister, [68](#)
- ~USBHandler
  - USBHandler, [123](#)
- AcceptConnections
  - Socket, [43](#)
- AddClient
  - Messenger, [37](#)
- Announce
  - Client, [17](#)
- Bind
  - Socket, [44](#)
- bit
  - TDCRegister, [68](#)
- Broadcast
  - Messenger, [37](#)
- CLIENT
  - Socket communication objects, [7](#)
- ChannelSelectError
  - HPTDC chip control, [11](#)
- Clear
  - TDCRegister, [68](#)
- Client, [15](#)
  - ~Client, [17](#)
  - Announce, [17](#)
  - Client, [17](#)
  - Connect, [18](#)
  - Disconnect, [18](#)
  - fClientId, [20](#)
  - flsConnected, [20](#)
  - GetType, [19](#)
  - ParseMessage, [19](#)
  - Receive, [19](#)
  - Send, [19](#)
- CloseFile
  - FPGAHandler, [26](#)
- CoarseError
  - HPTDC chip control, [11](#)
- config
  - file\_header\_t, [23](#)
- Configure
  - Socket, [44](#)
- Connect
  - Client, [18](#)
  - Messenger, [37](#)
- 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, [44](#)
- DETECTOR
  - Socket communication objects, [8](#)
- DLL\_160MHz
  - HPTDC chip control, [11](#)
- DLL\_320MHz
  - HPTDC chip control, [11](#)
- DLL\_40MHz
  - HPTDC chip control, [11](#)
- DLL\_Illegal
  - HPTDC chip control, [11](#)
- 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, [12](#)
- Decode
  - HTTPMessage, [31](#)
- Description
  - Exception, [21](#)
- DisableAllChannels
  - TDCControl, [58](#)
- DisableChannel
  - TDCControl, [58](#)
- Disconnect
  - Client, [18](#)
  - Messenger, [38](#)
- DisconnectClient
  - Messenger, [38](#)
- Dump
  - Exception, [21](#)
  - HTTPMessage, [31](#)
  - Message, [34](#)
  - SocketMessage, [52](#)
  - TDCControl, [59](#)
  - TDCSetup, [79](#)
- DumpConnected
  - Socket, [44](#)
- DumpDevice
  - USBHandler, [123](#)
- DumpRegister
  - TDCRegister, [68](#)
- E\_100ps
  - HPTDC chip control, [11](#)
- E\_12p5ns
  - HPTDC chip control, [11](#)
- E\_1p6ns
  - HPTDC chip control, [11](#)
- E\_200ps
  - HPTDC chip control, [11](#)
- E\_3p12ns
  - HPTDC chip control, [11](#)
- E\_400ps
  - HPTDC chip control, [11](#)
- E\_6p25ns
  - HPTDC chip control, [11](#)
- E\_800ps
  - HPTDC chip control, [11](#)
- EdgeResolution
  - HPTDC chip control, [11](#)
- EnableAllChannels
  - TDCControl, [59](#)
- EnableChannel
  - TDCControl, [59](#)
- EnablePattern
  - HPTDC chip control, [11](#)
- EnabledError
  - HPTDC chip control, [11](#)
- Encode
  - HTTPMessage, [31](#)
- Error
  - HPTDC chip control, [12](#)
- ErrorNumber
  - Exception, [22](#)
- ErrorState
  - FPGAHandler, [26](#)
- EventType
  - HPTDC chip control, [11](#)
- Exception, [20](#)
  - ~Exception, [21](#)
  - Description, [21](#)
  - Dump, [21](#)
  - ErrorNumber, [22](#)
  - Exception, [21](#)
  - fDescription, [22](#)
  - fErrorNumber, [22](#)
  - fFrom, [22](#)
  - fType, [22](#)
  - From, [22](#)
  - Type, [22](#)
  - TypeString, [22](#)
- fAddress
  - Socket, [47](#)
- fBuffer
  - Socket, [47](#)
- fClientId
  - Client, [20](#)
- fDescription
  - Exception, [22](#)
- fDevice
  - USBHandler, [124](#)
- fErrorNumber
  - Exception, [22](#)
- fFilename
  - FPGAHandler, [29](#)
- fFrom
  - Exception, [22](#)
- fHandle
  - USBHandler, [124](#)
- fIsConnected
  - Client, [20](#)
- fIsFileOpen
  - FPGAHandler, [29](#)
- fIsTDCInReadout
  - FPGAHandler, [29](#)
- fListenersInfo
  - Messenger, [41](#)

- fMaster
  - Socket, [47](#)
- fMessage
  - SocketMessage, [54](#)
- fNumAttempts
  - Messenger, [41](#)
- fNumWords
  - TDCRegister, [69](#)
- fOriginalString
  - HTTPMessage, [31](#)
- fOutput
  - FPGAHandler, [29](#)
- FPGAHandler, [24](#)
  - ~FPGAHandler, [26](#)
  - CloseFile, [26](#)
  - ErrorState, [26](#)
  - fFilename, [29](#)
  - fIsFileOpen, [29](#)
  - fIsTDCInReadout, [29](#)
  - fOutput, [29](#)
  - FPGAHandler, [26](#)
  - fTDCBSR, [29](#)
  - fTDCCControl, [29](#)
  - fTDCSetup, [29](#)
  - fTDCStatus, [29](#)
  - GetConfiguration, [26](#)
  - GetFilename, [26](#)
  - GetType, [27](#)
  - OpenFile, [27](#)
  - ReadBoundaryScanRegister, [27](#)
  - ReadBuffer, [27](#)
  - ReadConfiguration, [27](#)
  - ReadRegister, [27](#)
  - SendConfiguration, [27](#)
  - SetConfiguration, [28](#)
  - SetRegister, [28](#)
- fPort
  - Socket, [47](#)
- fReadFds
  - Socket, [47](#)
- fSocketId
  - Socket, [47](#)
- fSocketsConnected
  - Socket, [47](#)
- fString
  - Message, [34](#)
- fTDCBSR
  - FPGAHandler, [29](#)
- fTDCCControl
  - FPGAHandler, [29](#)
- fTDCSetup
  - FPGAHandler, [29](#)
- fTDCStatus
  - FPGAHandler, [29](#)
- fType
  - Exception, [22](#)
- fWS
  - HTTPMessage, [31](#)
- Messenger, [41](#)
- fWord
  - TDCEvent, [66](#)
  - TDCRegister, [69](#)
- fWordSize
  - TDCRegister, [69](#)
- Fetch
  - USBHandler, [123](#)
- FetchMessage
  - Socket, [44](#)
- file\_header\_t, [23](#)
  - config, [23](#)
  - magic, [23](#)
  - run\_id, [23](#)
  - spill\_id, [24](#)
- From
  - Exception, [22](#)
- GetBits
  - TDCRegister, [68](#)
- GetBunchId
  - TDCEvent, [64](#)
- GetChannelOffset
  - TDCSetup, [80](#)
- GetCoarseCountOffset
  - TDCSetup, [80](#)
- GetConfiguration
  - FPGAHandler, [26](#)
- GetDLLAdjustment
  - TDCSetup, [81](#)
- GetDLLReset
  - TDCCControl, [59](#)
- GetDeadTime
  - TDCSetup, [81](#)
- GetEdgeResolution
  - TDCSetup, [81](#)
- GetEdgesPairing
  - TDCSetup, [82](#)
- GetEnableError
  - TDCSetup, [82](#)
- GetEnableErrorBypass
  - TDCSetup, [82](#)
- GetEnableErrorMark
  - TDCSetup, [83](#)
- GetEnableJTAGReadout
  - TDCSetup, [83](#)
- GetEnablePattern
  - TDCCControl, [60](#)
- GetEnableReadoutOccupancy
  - TDCSetup, [83](#)
- GetEnableReadoutSeparator
  - TDCSetup, [84](#)
- GetEnableSerial
  - TDCSetup, [84](#)
- GetErrorFlags
  - TDCEvent, [64](#)
- GetEventId
  - TDCEvent, [64](#)
- GetFilename

- FPGAHandler, 26
- GetGlobalReset
  - TDCControl, 60
- GetIntValue
  - SocketMessage, 52
- GetKey
  - HTTPMessage, 31
  - Message, 34
  - SocketMessage, 52
- GetLeadingMode
  - TDCSetup, 84
- GetLeadingTime
  - TDCEvent, 65
- GetMatchWindow
  - TDCSetup, 85
- GetMaxEventSize
  - TDCSetup, 85
- GetNumWords
  - TDCRegister, 69
- GetPLLReset
  - TDCControl, 60
- GetPort
  - Socket, 44
- GetRCAdjustment
  - TDCSetup, 85
- GetReadoutFIFOSize
  - TDCSetup, 86
- GetRejectCountOffset
  - TDCSetup, 86
- GetRejectFIFOFull
  - TDCSetup, 86
- GetSearchWindow
  - TDCSetup, 87
- GetSetupParity
  - TDCSetup, 87
- GetSocketId
  - Socket, 45
- GetSocketType
  - Socket, 45
- GetString
  - Message, 34
  - SocketMessage, 52
- GetTDCId
  - TDCEvent, 65
- GetTestInvert
  - TDCSetup, 87
- GetTestMode
  - TDCSetup, 88
- GetTrailingMode
  - TDCSetup, 88
- GetTrailingTime
  - TDCEvent, 65
- GetTriggerCountOffset
  - TDCSetup, 88
- GetTriggerLatency
  - TDCSetup, 89
- GetTriggerMatchingMode
  - TDCSetup, 89
- GetType
  - Client, 19
  - FPGAHandler, 27
  - Messenger, 39
  - TDCEvent, 65
- GetValue
  - SocketMessage, 52
- GetVectorValue
  - SocketMessage, 52
- GetVernierOffset
  - TDCSetup, 89
- GetWidth
  - TDCEvent, 66
- GetWidthResolution
  - TDCSetup, 90
- GetWord
  - TDCRegister, 69
- GetWordCount
  - TDCEvent, 66
- 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, 11
  - DLL\_320MHz, 11
  - DLL\_40MHz, 11
  - DLL\_Illegal, 11
  - 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, 12
  - E\_100ps, 11
  - E\_12p5ns, 11
  - E\_1p6ns, 11
  - E\_200ps, 11
  - E\_3p12ns, 11
  - E\_400ps, 11
  - E\_6p25ns, 11
  - E\_800ps, 11
  - EdgeResolution, 11



- EnablePattern, 11
- EnabledError, 11
- Error, 12
- EventType, 11
- GroupHeader, 11
- GroupTrailer, 11
- IO\_aux\_clock, 12
- IO\_clock\_40, 12
- IO\_pll\_clock\_160, 12
- IO\_pll\_clock\_80, 12
- IOClockSource, 12
- Invalid, 11
- JTAGInstructionParityError, 11
- L1BufferParityError, 11
- LeadingEdge, 11
- R\_DLLReset, 12
- R\_EnablePattern, 12
- R\_GlobalReset, 12
- R\_PLLReset, 12
- RO\_Fixed, 12
- RO\_pll\_80Mbits\_s, 12
- RSC\_10Mbits\_s, 12
- RSC\_1p25Mbits\_s, 12
- RSC\_20Mbits\_s, 12
- RSC\_2p5Mbits\_s, 12
- RSC\_312p5kbits\_s, 12
- RSC\_40Mbits\_s, 12
- RSC\_5Mbits\_s, 12
- RSC\_625kbits\_s, 12
- ReadoutFIFOParityError, 11
- ReadoutSingleCycleSpeed, 12
- ReadoutSpeed, 12
- ReadoutStateError, 11
- RegisterName, 12
- SS\_DSStrobe, 13
- SS\_LeadingEdge, 13
- SS\_LeadingTrailingStrobe, 13
- SS\_NoStrobe, 13
- 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, 12
- TriggerFIFOParityError, 11
- TriggerMatchingError, 11
- VernierError, 11
- W\_100ns, 13
- W\_100ps, 13
- W\_12p5ns, 13
- W\_1p6ns, 13
- W\_200ns, 13
- W\_200ps, 13
- W\_25ns, 13
- W\_3p2ns, 13
- W\_400ns, 13
- W\_400ps, 13
- W\_50ns, 13
- W\_6p25ns, 13
- W\_800ns, 13
- W\_800ps, 13
- WidthResolution, 13
- HTTPMessage, 29
  - Decode, 31
  - Dump, 31
  - Encode, 31
  - fOriginalString, 31
  - fWS, 31
  - GetKey, 31
  - HTTPMessage, 30, 31
- INVALID
  - Socket communication objects, 7
- IO\_aux\_clock
  - HPTDC chip control, 12
- IO\_clock\_40
  - HPTDC chip control, 12
- IO\_pll\_clock\_160
  - HPTDC chip control, 12
- IO\_pll\_clock\_80
  - HPTDC chip control, 12
- IOClockSource
  - HPTDC chip control, 12
- Init
  - USBHandler, 123
- Invalid
  - HPTDC chip control, 11
- IsFromWeb
  - Message, 34
- IsWebSocket
  - Socket, 45
- JTAGInstructionParityError
  - HPTDC chip control, 11
- kCoarseCountOffset
  - TDCSetup, 117
- kControlParity
  - TDCControl, 62
- kCoreClockDelay
  - TDCSetup, 117
- kCoreClockSource
  - TDCSetup, 117
- kDLLClockDelay
  - TDCSetup, 117
- kDLLClockSource
  - TDCSetup, 117
- kDLLControl
  - TDCSetup, 117
- kDLLLock
  - TDCStatus, 122
- kDLLMode
  - TDCSetup, 117

- kDLLReset
  - TDCControl, [62](#)
- kDLLTapAdjust0
  - TDCSetup, [117](#)
- kDeadTime
  - TDCSetup, [117](#)
- kEnableAutomaticReject
  - TDCSetup, [117](#)
- kEnableBytewise
  - TDCSetup, [117](#)
- kEnableChannel
  - TDCControl, [62](#)
- kEnableDirectBunchReset
  - TDCSetup, [117](#)
- kEnableDirectEventReset
  - TDCSetup, [117](#)
- kEnableDirectTrigger
  - TDCSetup, [117](#)
- kEnableError
  - TDCSetup, [117](#)
- kEnableErrorBypass
  - TDCSetup, [117](#)
- kEnableErrorMark
  - TDCSetup, [117](#)
- kEnableGlobalHeader
  - TDCSetup, [117](#)
- kEnableGlobalTrailer
  - TDCSetup, [117](#)
- kEnableJTAGReadout
  - TDCSetup, [117](#)
- kEnableLocalHeader
  - TDCSetup, [117](#)
- kEnableLocalTrailer
  - TDCSetup, [117](#)
- kEnableMasterResetCode
  - TDCSetup, [118](#)
- kEnableMasterResetOnEventReset
  - TDCSetup, [118](#)
- kEnableMatching
  - TDCSetup, [118](#)
- kEnableOverflowDetect
  - TDCSetup, [118](#)
- kEnablePair
  - TDCSetup, [118](#)
- kEnablePattern
  - TDCControl, [62](#)
- kEnableReadoutOccupancy
  - TDCSetup, [118](#)
- kEnableReadoutSeparator
  - TDCSetup, [118](#)
- kEnableRelative
  - TDCSetup, [118](#)
- kEnableResetChannelBufferWhenSeparator
  - TDCSetup, [118](#)
- kEnableSeparatorOnBunchReset
  - TDCSetup, [118](#)
- kEnableSeparatorOnEventReset
  - TDCSetup, [118](#)
- kEnableSerial
  - TDCSetup, [118](#)
- kEnableSetCountersOnBunchReset
  - TDCSetup, [118](#)
- kEnableTTLClock
  - TDCSetup, [118](#)
- kEnableTTLControl
  - TDCSetup, [118](#)
- kEnableTTLHit
  - TDCSetup, [118](#)
- kEnableTTLReset
  - TDCSetup, [118](#)
- kEnableTTLSerial
  - TDCSetup, [118](#)
- kError
  - TDCStatus, [122](#)
- kEventCountOffset
  - TDCSetup, [118](#)
- kGlobalReset
  - TDCControl, [62](#)
- kHaveToken
  - TDCStatus, [122](#)
- kIOClockDelay
  - TDCSetup, [118](#)
- kIOClockSource
  - TDCSetup, [118](#)
- kKeepToken
  - TDCSetup, [118](#)
- kL1Occupancy
  - TDCStatus, [122](#)
- kLeading
  - TDCSetup, [118](#)
- kLeadingResolution
  - TDCSetup, [118](#)
- kLowPowerMode
  - TDCSetup, [118](#)
- kMaster
  - TDCSetup, [118](#)
- kMatchWindow
  - TDCSetup, [118](#)
- kMaxEventSize
  - TDCSetup, [118](#)
- kModeRC
  - TDCSetup, [119](#)
- kModeRCCompression
  - TDCSetup, [119](#)
- kOffset0
  - TDCSetup, [119](#)
- kPLLControl
  - TDCSetup, [119](#)
- kPLLReset
  - TDCControl, [62](#)
- kRCAdjust0
  - TDCSetup, [119](#)
- kReadoutFIFOEmpty
  - TDCStatus, [122](#)
- kReadoutFIFOFull
  - TDCStatus, [122](#)

- kReadoutFIFOOccupancy
  - TDCStatus, [122](#)
- kReadoutFIFOSize
  - TDCSetup, [119](#)
- kReadoutSingleCycleSpeed
  - TDCSetup, [119](#)
- kReadoutSpeedSelect
  - TDCSetup, [119](#)
- kRejectCountOffset
  - TDCSetup, [119](#)
- kRejectFIFOFull
  - TDCSetup, [119](#)
- kRollOver
  - TDCSetup, [119](#)
- kSearchWindow
  - TDCSetup, [119](#)
- kSelectBypassInputs
  - TDCSetup, [119](#)
- kSerialClockDelay
  - TDCSetup, [119](#)
- kSerialClockSource
  - TDCSetup, [119](#)
- kSerialDelay
  - TDCSetup, [119](#)
- kSetupParity
  - TDCSetup, [119](#)
- kStrobeSelect
  - TDCSetup, [119](#)
- kTDCId
  - TDCSetup, [119](#)
- kTestInvert
  - TDCSetup, [119](#)
- kTestMode
  - TDCSetup, [119](#)
- kTestSelect
  - TDCSetup, [119](#)
- kTokenDelay
  - TDCSetup, [119](#)
- kTrailing
  - TDCSetup, [119](#)
- kTriggerCountOffset
  - TDCSetup, [119](#)
- kTriggerFIFOEmpty
  - TDCStatus, [122](#)
- kTriggerFIFOFull
  - TDCStatus, [122](#)
- kTriggerFIFOOccupancy
  - TDCStatus, [122](#)
- kVernierOffset
  - TDCSetup, [119](#)
- kWidthSelect
  - TDCSetup, [119](#)
- L1BufferParityError
  - HPTDC chip control, [11](#)
- LeadingEdge
  - HPTDC chip control, [11](#)
- Listen
  - Socket, [45](#)
- ListenerInfo, [32](#)
  - name, [32](#)
  - type, [32](#)
- MASTER
  - Socket communication objects, [7](#)
- magic
  - file\_header\_t, [23](#)
- Message, [32](#)
  - ~Message, [33](#)
  - Dump, [34](#)
  - fString, [34](#)
  - GetKey, [34](#)
  - GetString, [34](#)
  - IsFromWeb, [34](#)
  - Message, [33](#)
- Messenger, [34](#)
  - ~Messenger, [36](#)
  - AddClient, [37](#)
  - Broadcast, [37](#)
  - Connect, [37](#)
  - Disconnect, [38](#)
  - DisconnectClient, [38](#)
  - fListenersInfo, [41](#)
  - fNumAttempts, [41](#)
  - fWS, [41](#)
  - GetType, [39](#)
  - Messenger, [36](#)
  - ProcessMessage, [39](#)
  - Receive, [39](#)
  - Send, [40](#)
  - SwitchClientType, [40](#)
- name
  - ListenerInfo, [32](#)
- Object
  - SocketMessage, [53](#)
- OpenFile
  - FPGAHandler, [27](#)
- ParseMessage
  - Client, [19](#)
- PrepareConnection
  - Socket, [45](#)
- ProcessMessage
  - Messenger, [39](#)
- R\_DLLReset
  - HPTDC chip control, [12](#)
- R\_EnablePattern
  - HPTDC chip control, [12](#)
- R\_GlobalReset
  - HPTDC chip control, [12](#)
- R\_PLLReset
  - HPTDC chip control, [12](#)
- RO\_Fixed
  - HPTDC chip control, [12](#)
- RO\_pll\_80Mbits\_s

- HPTDC chip control, [12](#)
- RSC\_10Mbits\_s
  - HPTDC chip control, [12](#)
- RSC\_1p25Mbits\_s
  - HPTDC chip control, [12](#)
- RSC\_20Mbits\_s
  - HPTDC chip control, [12](#)
- RSC\_2p5Mbits\_s
  - HPTDC chip control, [12](#)
- RSC\_312p5kbits\_s
  - HPTDC chip control, [12](#)
- RSC\_40Mbits\_s
  - HPTDC chip control, [12](#)
- RSC\_5Mbits\_s
  - HPTDC chip control, [12](#)
- RSC\_625kbits\_s
  - HPTDC chip control, [12](#)
- ReadBoundaryScanRegister
  - FPGAHandler, [27](#)
- ReadBuffer
  - FPGAHandler, [27](#)
- ReadConfiguration
  - FPGAHandler, [27](#)
- ReadRegister
  - FPGAHandler, [27](#)
- ReadoutFIFOParityError
  - HPTDC chip control, [11](#)
- ReadoutSingleCycleSpeed
  - HPTDC chip control, [12](#)
- ReadoutSpeed
  - HPTDC chip control, [12](#)
- ReadoutStateError
  - HPTDC chip control, [11](#)
- Receive
  - Client, [19](#)
  - Messenger, [39](#)
- RegisterName
  - HPTDC chip control, [12](#)
- run\_id
  - file\_header\_t, [23](#)
- SS\_DSStrobe
  - HPTDC chip control, [13](#)
- SS\_LeadingEdge
  - HPTDC chip control, [13](#)
- SS\_LeadingTrailingStrobe
  - HPTDC chip control, [13](#)
- SS\_NoStrobe
  - HPTDC chip control, [13](#)
- SelectConnections
  - Socket, [45](#)
- Send
  - Client, [19](#)
  - Messenger, [40](#)
- SendConfiguration
  - FPGAHandler, [27](#)
- SendMessage
  - Socket, [46](#)
- 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](#)
  - SerialClockSource
    - HPTDC chip control, [12](#)
  - SerialStrobeType
    - HPTDC chip control, [12](#)
  - SetAllChannelsOffset
    - TDCSetup, [90](#)
  - SetAllTapsDLLAdjustment
    - TDCSetup, [90](#)
  - SetBits
    - TDCRegister, [69](#)
  - SetBypassInputs
    - TDCSetup, [90](#)
  - SetChannelOffset
    - TDCSetup, [91](#)
  - SetCoarseCountOffset
    - TDCSetup, [91](#)
  - SetConfiguration
    - FPGAHandler, [28](#)
  - SetConstantValues
    - TDCBoundaryScanRegister, [56](#)
    - TDCControl, [61](#)
    - TDCRegister, [69](#)
    - TDCSetup, [91](#)
    - TDCStatus, [121](#)
  - SetControlParity
    - TDCControl, [61](#)
  - SetCoreClockDelay
    - TDCSetup, [92](#)
  - SetCoreClockSource
    - TDCSetup, [93](#)
  - SetDLLAdjustment
    - TDCSetup, [93](#)
  - SetDLLClockDelay
    - TDCSetup, [94](#)
  - SetDLLClockSource
    - TDCSetup, [94](#)
  - SetDLLControl
    - TDCSetup, [94](#)
  - SetDLLMode
    - TDCSetup, [95](#)
  - SetDLLReset
    - TDCControl, [61](#)
  - SetDeadTime
    - TDCSetup, [93](#)
  - SetEdgeResolution
    - TDCSetup, [95](#)
  - SetEdgesPairing
    - TDCSetup, [95](#)
  - SetEnableAutomaticReject
    - TDCSetup, [96](#)
  - SetEnableBytewise

- TDCSetup, 96
- SetEnableDirectBunchReset
  - TDCSetup, 96
- SetEnableDirectEventReset
  - TDCSetup, 97
- SetEnableDirectTrigger
  - TDCSetup, 97
- SetEnableError
  - TDCSetup, 97
- SetEnableErrorBypass
  - TDCSetup, 98
- SetEnableErrorMark
  - TDCSetup, 98
- SetEnableGlobalHeader
  - TDCSetup, 98
- SetEnableGlobalTrailer
  - TDCSetup, 99
- SetEnableJTAGReadout
  - TDCSetup, 99
- SetEnableLocalHeader
  - TDCSetup, 99
- SetEnableLocalTrailer
  - TDCSetup, 100
- SetEnableMasterResetCode
  - TDCSetup, 100
- SetEnableMasterResetOnEventReset
  - TDCSetup, 100
- SetEnableOverflowDetect
  - TDCSetup, 101
- SetEnablePattern
  - TDCControl, 62
- SetEnableReadoutOccupancy
  - TDCSetup, 101
- SetEnableReadoutSeparator
  - TDCSetup, 101
- SetEnableRelative
  - TDCSetup, 102
- SetEnableResetChannelBufferWhenSeparator
  - TDCSetup, 102
- SetEnableSeparatorOnBunchReset
  - TDCSetup, 102
- SetEnableSeparatorOnEventReset
  - TDCSetup, 103
- SetEnableSerial
  - TDCSetup, 103
- SetEnableSetCountersOnBunchReset
  - TDCSetup, 103
- SetEnableTTLClock
  - TDCSetup, 104
- SetEnableTTLControl
  - TDCSetup, 104
- SetEnableTTLHit
  - TDCSetup, 104
- SetEnableTTLReset
  - TDCSetup, 105
- SetEnableTTLSerial
  - TDCSetup, 105
- SetEventCountOffset
  - TDCSetup, 106
- SetGlobalReset
  - TDCControl, 62
- SetIOClockDelay
  - TDCSetup, 106
- SetIOClockSource
  - TDCSetup, 106
- SetKeepToken
  - TDCSetup, 107
- SetKeyValue
  - SocketMessage, 53, 54
- SetLeadingMode
  - TDCSetup, 107
- SetLowPowerMode
  - TDCSetup, 107
- SetMaster
  - TDCSetup, 108
- SetMatchWindow
  - TDCSetup, 108
- SetMaxEventSize
  - TDCSetup, 108
- SetModeRC
  - TDCSetup, 109
- SetModeRCCompression
  - TDCSetup, 109
- SetPLLControl
  - TDCSetup, 109
- SetPLLReset
  - TDCControl, 62
- SetPort
  - Socket, 46
- SetRCAdjustment
  - TDCSetup, 110
- SetReadoutFIFOSize
  - TDCSetup, 110
- SetReadoutSingleCycleSpeed
  - TDCSetup, 110
- SetReadoutSpeedSelect
  - TDCSetup, 111
- SetRegister
  - FPGAHandler, 28
- SetRejectCountOffset
  - TDCSetup, 111
- SetRejectFIFOFull
  - TDCSetup, 111
- SetRollOver
  - TDCSetup, 112
- SetSearchWindow
  - TDCSetup, 112
- SetSerialClockDelay
  - TDCSetup, 112
- SetSerialClockSource
  - TDCSetup, 113
- SetSerialDelay
  - TDCSetup, 113
- SetSetupParity
  - TDCSetup, 113
- SetSocketId

- Socket, 46
- SetStrobeSelect
  - TDCSetup, 114
- SetTestInvert
  - TDCSetup, 114
- SetTestMode
  - TDCSetup, 114
- SetTokenDelay
  - TDCSetup, 115
- SetTrailingMode
  - TDCSetup, 115
- SetTriggerCountOffset
  - TDCSetup, 115
- SetTriggerMatchingMode
  - TDCSetup, 116
- SetVernierOffset
  - TDCSetup, 116
- SetWidthResolution
  - TDCSetup, 116
- SetWord
  - TDCRegister, 69
- SetupParityError
  - HPTDC chip control, 11
- Socket, 41
  - ~Socket, 43
  - AcceptConnections, 43
  - Bind, 44
  - Configure, 44
  - Create, 44
  - DumpConnected, 44
  - fAddress, 47
  - fBuffer, 47
  - fMaster, 47
  - fPort, 47
  - fReadFds, 47
  - fSocketId, 47
  - fSocketsConnected, 47
  - FetchMessage, 44
  - GetPort, 44
  - GetSocketId, 45
  - GetSocketType, 45
  - IsWebSocket, 45
  - Listen, 45
  - PrepareConnection, 45
  - SelectConnections, 45
  - SendMessage, 46
  - SetPort, 46
  - SetSocketId, 46
  - Socket, 43
  - SocketCollection, 43
  - Start, 46
  - Stop, 46
- Socket communication objects, 7
  - CLIENT, 7
  - DETECTOR, 8
  - INVALID, 7
  - MASTER, 7
  - SocketType, 7
  - WEBSOCKET\_CLIENT, 7
- SocketCollection
  - Socket, 43
- SocketMessage, 47
  - ~SocketMessage, 52
  - Dump, 52
  - fMessage, 54
  - GetIntValue, 52
  - GetKey, 52
  - GetString, 52
  - GetValue, 52
  - GetVectorValue, 52
  - Object, 53
  - SetKeyValue, 53, 54
  - SocketMessage, 49–51
  - String, 54
- SocketType
  - Socket communication objects, 7
- spill\_id
  - file\_header\_t, 24
- Start
  - Socket, 46
- Stop
  - Socket, 46
- String
  - SocketMessage, 54
- SwitchClientType
  - Messenger, 40
- TDCBoundaryScanRegister, 54
  - SetConstantValues, 56
  - TDCBoundaryScanRegister, 55
- TDCCControl, 56
  - DisableAllChannels, 58
  - DisableChannel, 58
  - Dump, 59
  - EnableAllChannels, 59
  - EnableChannel, 59
  - GetDLLReset, 59
  - GetEnablePattern, 60
  - GetGlobalReset, 60
  - GetPLLReset, 60
  - kControlParity, 62
  - kDLLReset, 62
  - kEnableChannel, 62
  - kEnablePattern, 62
  - kGlobalReset, 62
  - kPLLReset, 62
  - SetConstantValues, 61
  - SetControlParity, 61
  - SetDLLReset, 61
  - SetEnablePattern, 62
  - SetGlobalReset, 62
  - SetPLLReset, 62
  - TDCCControl, 58
- TDCEvent, 63
  - ~TDCEvent, 64
  - fWord, 66
  - GetBunchId, 64

- GetErrorFlags, [64](#)
- GetEventId, [64](#)
- GetLeadingTime, [65](#)
- GetTDCId, [65](#)
- GetTrailingTime, [65](#)
- GetType, [65](#)
- GetWidth, [66](#)
- GetWordCount, [66](#)
- TDCEvent, [64](#)
- TDCHeader
  - HPTDC chip control, [11](#)
- TDCRegister, [66](#)
  - ~TDCRegister, [68](#)
  - bit, [68](#)
  - Clear, [68](#)
  - DumpRegister, [68](#)
  - fNumWords, [69](#)
  - fWord, [69](#)
  - fWordSize, [69](#)
  - GetBits, [68](#)
  - GetNumWords, [69](#)
  - GetWord, [69](#)
  - SetBits, [69](#)
  - SetConstantValues, [69](#)
  - SetWord, [69](#)
  - TDCRegister, [68](#)
  - word\_t, [68](#)
- TDCSetup, [70](#)
  - Dump, [79](#)
  - GetChannelOffset, [80](#)
  - GetCoarseCountOffset, [80](#)
  - GetDLLAdjustment, [81](#)
  - GetDeadTime, [81](#)
  - GetEdgeResolution, [81](#)
  - GetEdgesPairing, [82](#)
  - GetEnableError, [82](#)
  - GetEnableErrorBypass, [82](#)
  - GetEnableErrorMark, [83](#)
  - GetEnableJTAGReadout, [83](#)
  - GetEnableReadoutOccupancy, [83](#)
  - GetEnableReadoutSeparator, [84](#)
  - GetEnableSerial, [84](#)
  - GetLeadingMode, [84](#)
  - GetMatchWindow, [85](#)
  - GetMaxEventSize, [85](#)
  - GetRCAdjustment, [85](#)
  - GetReadoutFIFOSize, [86](#)
  - GetRejectCountOffset, [86](#)
  - GetRejectFIFOFull, [86](#)
  - GetSearchWindow, [87](#)
  - GetSetupParity, [87](#)
  - GetTestInvert, [87](#)
  - GetTestMode, [88](#)
  - GetTrailingMode, [88](#)
  - GetTriggerCountOffset, [88](#)
  - GetTriggerLatency, [89](#)
  - GetTriggerMatchingMode, [89](#)
  - GetVernierOffset, [89](#)
  - GetWidthResolution, [90](#)
  - kCoarseCountOffset, [117](#)
  - kCoreClockDelay, [117](#)
  - kCoreClockSource, [117](#)
  - kDLLClockDelay, [117](#)
  - kDLLClockSource, [117](#)
  - kDLLControl, [117](#)
  - kDLLMode, [117](#)
  - kDLLTapAdjust0, [117](#)
  - kDeadTime, [117](#)
  - kEnableAutomaticReject, [117](#)
  - kEnableBytewise, [117](#)
  - kEnableDirectBunchReset, [117](#)
  - kEnableDirectEventReset, [117](#)
  - kEnableDirectTrigger, [117](#)
  - kEnableError, [117](#)
  - kEnableErrorBypass, [117](#)
  - kEnableErrorMark, [117](#)
  - kEnableGlobalHeader, [117](#)
  - kEnableGlobalTrailer, [117](#)
  - kEnableJTAGReadout, [117](#)
  - kEnableLocalHeader, [117](#)
  - kEnableLocalTrailer, [117](#)
  - kEnableMasterResetCode, [118](#)
  - kEnableMasterResetOnEventReset, [118](#)
  - kEnableMatching, [118](#)
  - kEnableOverflowDetect, [118](#)
  - kEnablePair, [118](#)
  - kEnableReadoutOccupancy, [118](#)
  - kEnableReadoutSeparator, [118](#)
  - kEnableRelative, [118](#)
  - kEnableResetChannelBufferWhenSeparator, [118](#)
  - kEnableSeparatorOnBunchReset, [118](#)
  - kEnableSeparatorOnEventReset, [118](#)
  - kEnableSerial, [118](#)
  - kEnableSetCountersOnBunchReset, [118](#)
  - kEnableTTLClock, [118](#)
  - kEnableTTLControl, [118](#)
  - kEnableTTLHit, [118](#)
  - kEnableTTLReset, [118](#)
  - kEnableTTLSerial, [118](#)
  - kEventCountOffset, [118](#)
  - kIOClockDelay, [118](#)
  - kIOClockSource, [118](#)
  - kKeepToken, [118](#)
  - kLeading, [118](#)
  - kLeadingResolution, [118](#)
  - kLowPowerMode, [118](#)
  - kMaster, [118](#)
  - kMatchWindow, [118](#)
  - kMaxEventSize, [118](#)
  - kModeRC, [119](#)
  - kModeRCCompression, [119](#)
  - kOffset0, [119](#)
  - kPLLControl, [119](#)
  - kRCAdjust0, [119](#)
  - kReadoutFIFOSize, [119](#)
  - kReadoutSingleCycleSpeed, [119](#)

- kReadoutSpeedSelect, 119
- kRejectCountOffset, 119
- kRejectFIFOFull, 119
- kRollOver, 119
- kSearchWindow, 119
- kSelectBypassInputs, 119
- kSerialClockDelay, 119
- kSerialClockSource, 119
- kSerialDelay, 119
- kSetupParity, 119
- kStrobeSelect, 119
- kTDCId, 119
- kTestInvert, 119
- kTestMode, 119
- kTestSelect, 119
- kTokenDelay, 119
- kTrailing, 119
- kTriggerCountOffset, 119
- kVernierOffset, 119
- kWidthSelect, 119
- SetAllChannelsOffset, 90
- SetAllTapsDLLAdjustment, 90
- SetBypassInputs, 90
- SetChannelOffset, 91
- SetCoarseCountOffset, 91
- SetConstantValues, 91
- SetCoreClockDelay, 92
- SetCoreClockSource, 93
- SetDLLAdjustment, 93
- SetDLLClockDelay, 94
- SetDLLClockSource, 94
- SetDLLControl, 94
- SetDLLMode, 95
- SetDeadTime, 93
- SetEdgeResolution, 95
- SetEdgesPairing, 95
- SetEnableAutomaticReject, 96
- SetEnableBytewise, 96
- SetEnableDirectBunchReset, 96
- SetEnableDirectEventReset, 97
- SetEnableDirectTrigger, 97
- SetEnableError, 97
- SetEnableErrorBypass, 98
- SetEnableErrorMark, 98
- SetEnableGlobalHeader, 98
- SetEnableGlobalTrailer, 99
- SetEnableJTAGReadout, 99
- SetEnableLocalHeader, 99
- SetEnableLocalTrailer, 100
- SetEnableMasterResetCode, 100
- SetEnableMasterResetOnEventReset, 100
- SetEnableOverflowDetect, 101
- SetEnableReadoutOccupancy, 101
- SetEnableReadoutSeparator, 101
- SetEnableRelative, 102
- SetEnableResetChannelBufferWhenSeparator, 102
- SetEnableSeparatorOnBunchReset, 102
- SetEnableSeparatorOnEventReset, 103
- SetEnableSerial, 103
- SetEnableSetCountersOnBunchReset, 103
- SetEnableTTLClock, 104
- SetEnableTTLControl, 104
- SetEnableTTLHit, 104
- SetEnableTTLReset, 105
- SetEnableTTLSerial, 105
- SetEventCountOffset, 106
- SetIOClockDelay, 106
- SetIOClockSource, 106
- SetKeepToken, 107
- SetLeadingMode, 107
- SetLowPowerMode, 107
- SetMaster, 108
- SetMatchWindow, 108
- SetMaxEventSize, 108
- SetModeRC, 109
- SetModeRCCompression, 109
- SetPLLControl, 109
- SetRCAdjustment, 110
- SetReadoutFIFOSize, 110
- SetReadoutSingleCycleSpeed, 110
- SetReadoutSpeedSelect, 111
- SetRejectCountOffset, 111
- SetRejectFIFOFull, 111
- SetRollOver, 112
- SetSearchWindow, 112
- SetSerialClockDelay, 112
- SetSerialClockSource, 113
- SetSerialDelay, 113
- SetSetupParity, 113
- SetStrobeSelect, 114
- SetTestInvert, 114
- SetTestMode, 114
- SetTokenDelay, 115
- SetTrailingMode, 115
- SetTriggerCountOffset, 115
- SetTriggerMatchingMode, 116
- SetVernierOffset, 116
- SetWidthResolution, 116
- TDCSetup, 77, 78
- TDCStatus, 120
  - kDLLLock, 122
  - kError, 122
  - kHaveToken, 122
  - kL1Occupancy, 122
  - kReadoutFIFOEmpty, 122
  - kReadoutFIFOFull, 122
  - kReadoutFIFOOccupancy, 122
  - kTriggerFIFOEmpty, 122
  - kTriggerFIFOFull, 122
  - kTriggerFIFOOccupancy, 122
  - SetConstantValues, 121
  - TDCStatus, 121
- TDCTrailer
  - HPTDC chip control, 11
- TrailingEdge



- HPTDC chip control, [12](#)
- TriggerFIFOParityError
  - HPTDC chip control, [11](#)
- TriggerMatchingError
  - HPTDC chip control, [11](#)
- Type
  - Exception, [22](#)
- type
  - ListenerInfo, [32](#)
- TypeString
  - Exception, [22](#)
- USBHandler, [122](#)
  - ~USBHandler, [123](#)
  - DumpDevice, [123](#)
  - fDevice, [124](#)
  - fHandle, [124](#)
  - Fetch, [123](#)
  - Init, [123](#)
  - USBHandler, [123](#)
  - Write, [124](#)
- VernierError
  - HPTDC chip control, [11](#)
- W\_100ns
  - HPTDC chip control, [13](#)
- W\_100ps
  - HPTDC chip control, [13](#)
- W\_12p5ns
  - HPTDC chip control, [13](#)
- W\_1p6ns
  - HPTDC chip control, [13](#)
- W\_200ns
  - HPTDC chip control, [13](#)
- W\_200ps
  - HPTDC chip control, [13](#)
- W\_25ns
  - HPTDC chip control, [13](#)
- W\_3p2ns
  - HPTDC chip control, [13](#)
- W\_400ns
  - HPTDC chip control, [13](#)
- W\_400ps
  - HPTDC chip control, [13](#)
- W\_50ns
  - HPTDC chip control, [13](#)
- W\_6p25ns
  - HPTDC chip control, [13](#)
- W\_800ns
  - HPTDC chip control, [13](#)
- W\_800ps
  - HPTDC chip control, [13](#)
- WEBSOCKET\_CLIENT
  - Socket communication objects, [7](#)
- WidthResolution
  - HPTDC chip control, [13](#)
- word\_t
  - TDCRegister, [68](#)
- Write
  - USBHandler, [124](#)