2015 Test beam Run Control

Generated by Doxygen 1.8.9.1

Wed Jun 3 2015 17:39:49

Contents

1 Module Index **1**

1.1 Modules . . . 1

2 Hierarchical Index **3**

2.1 Class Hierarchy . . . 3

3 Data Structure Index **5**

3.1 Data Structures . . . 5

4 Module Documentation **7**

4.1 FPGA board control . . . 7

4.1.1 Detailed Description . . . 7

4.2 Socket communication objects . . . 8

4.2.1 Detailed Description . . . 8

4.2.2 Enumeration Type Documentation . . . 8

4.2.2.1 SocketType . . . 8

4.3 HPTDC chip control . . . 9

4.3.1 Detailed Description . . . 10

4.3.2 Enumeration Type Documentation . . . 10

4.3.2.1 CoreClockSource . . . 10

4.3.2.2 DeadTime . . . 10

4.3.2.3 DLLClockSource . . . 10

4.3.2.4 DLLSpeedMode . . . 10

4.3.2.5 EdgeResolution . . . 11

4.3.2.6 EnabledError . . . 11

4.3.2.7 EnablePattern . . . 11

4.3.2.8 EventType . . . 11

4.3.2.9 IOClockSource . . . 12

4.3.2.10 ReadoutSingleCycleSpeed . . . 12

4.3.2.11 ReadoutSpeed . . . 12

4.3.2.12 RegisterName . . . 12

4.3.2.13 SerialClockSource . . . 12

4.3.2.14 SerialStrobeType . . . 13
4.3.2.15 WidthResolution . . . 13

5 Data Structure Documentation **15**

5.1 Client Class Reference . . . 15
5.1.1 Detailed Description . . . 17
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
5.1.3.8 SendAndReceive . . . 20
5.1.4 Field Documentation . . . 20
5.1.4.1 fClientId . . . 20
5.1.4.2 fIsConnected . . . 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 . . . 22
5.2.3.1 Description . . . 22
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 . . . 23

5.3 file_header_t Struct Reference . . . 23
5.3.1 Detailed Description . . . 23
5.3.2 Field Documentation . . . 23
5.3.2.1 magic . . . 23
5.3.2.2 num_hptdc . . . 23
5.3.2.3 run_id . . . 23
5.3.2.4 spill_id . . . 23
5.4 FileReader Class Reference . . . 23
5.4.1 Constructor & Destructor Documentation . . . 24
5.4.1.1 FileReader . . . 25
5.4.1.2 ~FileReader . . . 25
5.4.2 Member Function Documentation . . . 25
5.4.2.1 GetNextEvent . . . 25
5.4.2.2 GetNumTDCs . . . 25
5.4.3 Field Documentation . . . 25
5.4.3.1 fFile . . . 25
5.4.3.2 fHeader . . . 25
5.4.3.3 fSetupCollection . . . 25
5.5 FPGAHandler Class Reference . . . 25
5.5.1 Detailed Description . . . 27
5.5.2 Constructor & Destructor Documentation . . . 27
5.5.2.1 FPGAHandler . . . 27
5.5.2.2 ~FPGAHandler . . . 28
5.5.3 Member Function Documentation . . . 28
5.5.3.1 CloseFile . . . 28
5.5.3.2 ErrorState . . . 28
5.5.3.3 GetFilename . . . 28
5.5.3.4 GetTDC . . . 28
5.5.3.5 GetType . . . 28
5.5.3.6 OpenFile . . . 28
5.5.3.7 ReadBuffer . . . 29
5.5.3.8 SetTDCSetup . . . 29
5.5.3.9 Stop . . . 29
5.5.4 Field Documentation . . . 29
5.5.4.1 fFilename . . . 29
5.5.4.2 fIsFileOpen . . . 29
5.5.4.3 fIsTDCInReadout . . . 29
5.5.4.4 fOutput . . . 29
5.5.4.5 fTDC . . . 29
5.6 HTTPMessage Class Reference . . . 29

5.6.1 Detailed Description . . . 30
5.6.2 Constructor & Destructor Documentation . . . 31
5.6.2.1 HTTPMessage . . . 31
5.6.2.2 HTTPMessage . . . 31
5.6.3 Member Function Documentation . . . 31
5.6.3.1 Decode . . . 31
5.6.3.2 Dump . . . 31
5.6.3.3 Encode . . . 31
5.6.3.4 GetKey . . . 32
5.6.4 Field Documentation . . . 32
5.6.4.1 fOriginalString . . . 32
5.6.4.2 fWS . . . 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 . . . 33
5.7.3 Member Function Documentation . . . 33
5.7.3.1 Dump . . . 33
5.7.3.2 GetKey . . . 33
5.7.3.3 GetString . . . 33
5.7.3.4 IsFromWeb . . . 33
5.7.4 Field Documentation . . . 33
5.7.4.1 fString . . . 33
5.8 Messenger Class Reference . . . 34
5.8.1 Detailed Description . . . 35
5.8.2 Constructor & Destructor Documentation . . . 35
5.8.2.1 Messenger . . . 35
5.8.2.2 Messenger . . . 35
5.8.2.3 ∼Messenger . . . 36
5.8.3 Member Function Documentation . . . 36
5.8.3.1 AddClient . . . 36
5.8.3.2 Broadcast . . . 37
5.8.3.3 Connect . . . 37
5.8.3.4 Disconnect . . . 37
5.8.3.5 DisconnectClient . . . 38
5.8.3.6 GetType . . . 38
5.8.3.7 ProcessMessage . . . 38

5.8.3.8 Receive . . . 39
5.8.3.9 Send . . . 39
5.8.3.10 StartAcquisition . . . 40
5.8.3.11 StopAcquisition . . . 40
5.8.3.12 SwitchClientType . . . 41
5.8.4 Field Documentation . . . 41
5.8.4.1 fNumAttempts . . . 41
5.8.4.2 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 TDC Class Reference . . . 54
5.11.1 Detailed Description . . . 55
5.11.2 Constructor & Destructor Documentation . . . 55
5.11.2.1 TDC . . . 55
5.11.2.2 ∼TDC . . . 56
5.11.3 Member Function Documentation . . . 56
5.11.3.1 CheckFirmwareVersion . . . 56

5.11.3.2 FetchEvents . . . 56
5.11.3.3 GetSetupRegister . . . 56
5.11.3.4 ReadConfiguration . . . 56
5.11.3.5 ReadRegister . . . 56
5.11.3.6 ReadStatus . . . 57
5.11.3.7 SendConfiguration . . . 57
5.11.3.8 SetSetupRegister . . . 57
5.11.3.9 SoftReset . . . 57
5.11.3.10 WriteRegister . . . 57
5.11.4 Field Documentation . . . 58
5.11.4.1 fBS . . . 58
5.11.4.2 fControl . . . 58
5.11.4.3 fId . . . 58
5.11.4.4 fSetup . . . 58
5.11.4.5 fStatus . . . 58
5.11.4.6 fUSB . . . 58
5.12 TDCBoundaryScan Class Reference . . . 58
5.12.1 Detailed Description . . . 59
5.12.2 Constructor & Destructor Documentation . . . 60
5.12.2.1 TDCBoundaryScan . . . 60
5.12.2.2 TDCBoundaryScan . . . 60
5.12.3 Member Function Documentation . . . 60
5.12.3.1 SetConstantValues . . . 60
5.12.4 Field Documentation . . . 60
5.12.4.1 kAuxClock . . . 60
5.12.4.2 kBunchReset . . . 60
5.12.4.3 kClk . . . 60
5.12.4.4 kDataReady . . . 60
5.12.4.5 kEncodedControl . . . 60
5.12.4.6 kError . . . 60
5.12.4.7 kEventReset . . . 61
5.12.4.8 kGetData . . . 61
5.12.4.9 kHit . . . 61
5.12.4.10 kParallelDataOut . . . 61
5.12.4.11 kParallelEnable . . . 61
5.12.4.12 kReset . . . 61
5.12.4.13 kSerialBypassIn . . . 61
5.12.4.14 kSerialIn . . . 61
5.12.4.15 kSerialOut . . . 61
5.12.4.16 kStrobeOut . . . 61

5.12.4.17 kTest . . . 61
5.12.4.18 kTokenBypassIn . . . 61
5.12.4.19 kTokenIn . . . 61
5.12.4.20 kTokenOut . . . 61
5.12.4.21 kTrigger . . . 61
5.13 TDCControl Class Reference . . . 61
5.13.1 Detailed Description . . . 63
5.13.2 Constructor & Destructor Documentation . . . 63
5.13.2.1 TDCControl . . . 63
5.13.2.2 TDCControl . . . 63
5.13.3 Member Function Documentation . . . 63
5.13.3.1 DisableAllChannels . . . 63
5.13.3.2 DisableChannel . . . 64
5.13.3.3 Dump . . . 64
5.13.3.4 EnableAllChannels . . . 64
5.13.3.5 EnableChannel . . . 64
5.13.3.6 GetDLLReset . . . 65
5.13.3.7 GetEnablePattern . . . 65
5.13.3.8 GetGlobalReset . . . 65
5.13.3.9 GetPLLReset . . . 66
5.13.3.10 SetConstantValues . . . 66
5.13.3.11 SetControlParity . . . 66
5.13.3.12 SetDLLReset . . . 66
5.13.3.13 SetEnablePattern . . . 67
5.13.3.14 SetGlobalReset . . . 67
5.13.3.15 SetPLLReset . . . 67
5.13.4 Field Documentation . . . 67
5.13.4.1 kControlParity . . . 67
5.13.4.2 kDLLReset . . . 67
5.13.4.3 kEnableChannel . . . 67
5.13.4.4 kEnablePattern . . . 67
5.13.4.5 kGlobalReset . . . 67
5.13.4.6 kPLLReset . . . 68
5.14 TDCEvent Class Reference . . . 68
5.14.1 Detailed Description . . . 69
5.14.2 Constructor & Destructor Documentation . . . 69
5.14.2.1 TDCEvent . . . 69
5.14.2.2 TDCEvent . . . 69
5.14.2.3 ~TDCEvent . . . 69
5.14.3 Member Function Documentation . . . 69

5.14.3.1 GetBunchId . . . 69
5.14.3.2 GetErrorFlags . . . 69
5.14.3.3 GetEventId . . . 70
5.14.3.4 GetLeadingTime . . . 70
5.14.3.5 GetTDCId . . . 70
5.14.3.6 GetTrailingTime . . . 70
5.14.3.7 GetType . . . 71
5.14.3.8 GetWidth . . . 71
5.14.3.9 GetWordCount . . . 71
5.14.3.10 SetWord . . . 71
5.14.4 Field Documentation . . . 71
5.14.4.1 fWord . . . 71
5.15 TDCRegister Class Reference . . . 71
5.15.1 Detailed Description . . . 73
5.15.2 Member Typedef Documentation . . . 73
5.15.2.1 bit . . . 73
5.15.2.2 word_t . . . 73
5.15.3 Constructor & Destructor Documentation . . . 73
5.15.3.1 TDCRegister . . . 73
5.15.3.2 TDCRegister . . . 73
5.15.3.3 ~TDCRegister . . . 74
5.15.4 Member Function Documentation . . . 74
5.15.4.1 Clear . . . 74
5.15.4.2 DumpRegister . . . 74
5.15.4.3 GetBits . . . 74
5.15.4.4 GetNumWords . . . 74
5.15.4.5 GetWord . . . 74
5.15.4.6 operator= . . . 74
5.15.4.7 SetBits . . . 74
5.15.4.8 SetConstantValues . . . 74
5.15.4.9 SetWord . . . 74
5.15.5 Field Documentation . . . 75
5.15.5.1 fNumWords . . . 75
5.15.5.2 fWord . . . 75
5.15.5.3 fWordSize . . . 75
5.16 TDCSetup Class Reference . . . 75
5.16.1 Detailed Description . . . 82
5.16.2 Constructor & Destructor Documentation . . . 82
5.16.2.1 TDCSetup . . . 83
5.16.2.2 TDCSetup . . . 84

5.16.3 Member Function Documentation . . . 84
5.16.3.1 Dump . . . 85
5.16.3.2 GetChannelOffset . . . 85
5.16.3.3 GetCoarseCountOffset . . . 85
5.16.3.4 GetDeadTime . . . 86
5.16.3.5 GetDLLAdjustment . . . 86
5.16.3.6 GetEdgeResolution . . . 87
5.16.3.7 GetEdgesPairing . . . 87
5.16.3.8 GetEnableError . . . 87
5.16.3.9 GetEnableErrorBypass . . . 88
5.16.3.10 GetEnableErrorMark . . . 88
5.16.3.11 GetEnableJTAGReadout . . . 88
5.16.3.12 GetEnableReadoutOccupancy . . . 89
5.16.3.13 GetEnableReadoutSeparator . . . 89
5.16.3.14 GetEnableSerial . . . 89
5.16.3.15 GetLeadingMode . . . 89
5.16.3.16 GetMatchWindow . . . 90
5.16.3.17 GetMaxEventSize . . . 90
5.16.3.18 GetRCAdjustment . . . 90
5.16.3.19 GetReadoutFIFOSize . . . 91
5.16.3.20 GetRejectCountOffset . . . 91
5.16.3.21 GetRejectFIFOFull . . . 91
5.16.3.22 GetSearchWindow . . . 92
5.16.3.23 GetSetupParity . . . 92
5.16.3.24 GetTestInvert . . . 93
5.16.3.25 GetTestMode . . . 93
5.16.3.26 GetTrailingMode . . . 93
5.16.3.27 GetTriggerCountOffset . . . 93
5.16.3.28 GetTriggerLatency . . . 94
5.16.3.29 GetTriggerMatchingMode . . . 94
5.16.3.30 GetVernierOffset . . . 94
5.16.3.31 GetWidthResolution . . . 95
5.16.3.32 SetAllChannelsOffset . . . 95
5.16.3.33 SetAllTapsDLLAdjustment . . . 95
5.16.3.34 SetBypassInputs . . . 96
5.16.3.35 SetChannelOffset . . . 96
5.16.3.36 SetCoarseCountOffset . . . 96
5.16.3.37 SetConstantValues . . . 96
5.16.3.38 SetCoreClockDelay . . . 97
5.16.3.39 SetCoreClockSource . . . 98

5.16.3.40 SetDeadTime . . . 98
5.16.3.41 SetDLLAdjustment . . . 98
5.16.3.42 SetDLLClockDelay . . . 99
5.16.3.43 SetDLLClockSource . . . 99
5.16.3.44 SetDLLControl . . . 99
5.16.3.45 SetDLLMode . . . 100
5.16.3.46 SetEdgeResolution . . . 100
5.16.3.47 SetEdgesPairing . . . 100
5.16.3.48 SetEnableAutomaticReject . . . 101
5.16.3.49 SetEnableBytewise . . . 101
5.16.3.50 SetEnableDirectBunchReset . . . 101
5.16.3.51 SetEnableDirectEventReset . . . 102
5.16.3.52 SetEnableDirectTrigger . . . 102
5.16.3.53 SetEnableError . . . 102
5.16.3.54 SetEnableErrorBypass . . . 103
5.16.3.55 SetEnableErrorMark . . . 103
5.16.3.56 SetEnableGlobalHeader . . . 103
5.16.3.57 SetEnableGlobalTrailer . . . 104
5.16.3.58 SetEnableJTAGReadout . . . 104
5.16.3.59 SetEnableLocalHeader . . . 104
5.16.3.60 SetEnableLocalTrailer . . . 105
5.16.3.61 SetEnableMasterResetCode . . . 105
5.16.3.62 SetEnableMasterResetOnEventReset . . . 105
5.16.3.63 SetEnableOverflowDetect . . . 106
5.16.3.64 SetEnableReadoutOccupancy . . . 106
5.16.3.65 SetEnableReadoutSeparator . . . 106
5.16.3.66 SetEnableRelative . . . 107
5.16.3.67 SetEnableResetChannelBufferWhenSeparator . . . 107
5.16.3.68 SetEnableSeparatorOnBunchReset . . . 107
5.16.3.69 SetEnableSeparatorOnEventReset . . . 108
5.16.3.70 SetEnableSerial . . . 108
5.16.3.71 SetEnableSetCountersOnBunchReset . . . 108
5.16.3.72 SetEnableTTLClock . . . 109
5.16.3.73 SetEnableTTLControl . . . 109
5.16.3.74 SetEnableTTLHit . . . 110
5.16.3.75 SetEnableTTLReset . . . 110
5.16.3.76 SetEnableTTLSerial . . . 110
5.16.3.77 SetEventCountOffset . . . 111
5.16.3.78 SetIOClockDelay . . . 111
5.16.3.79 SetIOClockSource . . . 111

5.16.3.80 SetKeepToken . . . 112
5.16.3.81 SetLeadingMode . . . 112
5.16.3.82 SetLowPowerMode . . . 112
5.16.3.83 SetMaster . . . 113
5.16.3.84 SetMatchWindow . . . 113
5.16.3.85 SetMaxEventSize . . . 113
5.16.3.86 SetModeRC . . . 114
5.16.3.87 SetModeRCCompression . . . 114
5.16.3.88 SetPLLControl . . . 114
5.16.3.89 SetRCAdjustment . . . 115
5.16.3.90 SetReadoutFIFOSize . . . 115
5.16.3.91 SetReadoutSingleCycleSpeed . . . 115
5.16.3.92 SetReadoutSpeedSelect . . . 116
5.16.3.93 SetRejectCountOffset . . . 116
5.16.3.94 SetRejectFIFOFull . . . 117
5.16.3.95 SetRollOver . . . 117
5.16.3.96 SetSearchWindow . . . 117
5.16.3.97 SetSerialClockDelay . . . 117
5.16.3.98 SetSerialClockSource . . . 118
5.16.3.99 SetSerialDelay . . . 118
5.16.3.100 SetSetupParity . . . 118
5.16.3.101 SetStrobeSelect . . . 119
5.16.3.102 SetTestInvert . . . 119
5.16.3.103 SetTestMode . . . 119
5.16.3.104 SetTokenDelay . . . 120
5.16.3.105 SetTrailingMode . . . 120
5.16.3.106 SetTriggerCountOffset . . . 120
5.16.3.107 SetTriggerMatchingMode . . . 121
5.16.3.108 SetVernierOffset . . . 121
5.16.3.109 SetWidthResolution . . . 121
5.16.4 Field Documentation . . . 122
5.16.4.1 kCoarseCountOffset . . . 122
5.16.4.2 kCoreClockDelay . . . 122
5.16.4.3 kCoreClockSource . . . 122
5.16.4.4 kDeadTime . . . 122
5.16.4.5 kDLLClockDelay . . . 122
5.16.4.6 kDLLClockSource . . . 122
5.16.4.7 kDLLControl . . . 122
5.16.4.8 kDLLMode . . . 122
5.16.4.9 kDLLTapAdjust0 . . . 122

5.16.4.10 kEnableAutomaticReject . . . 122
5.16.4.11 kEnableBytewise . . . 122
5.16.4.12 kEnableDirectBunchReset . . . 122
5.16.4.13 kEnableDirectEventReset . . . 122
5.16.4.14 kEnableDirectTrigger . . . 122
5.16.4.15 kEnableError . . . 122
5.16.4.16 kEnableErrorBypass . . . 122
5.16.4.17 kEnableErrorMark . . . 122
5.16.4.18 kEnableGlobalHeader . . . 122
5.16.4.19 kEnableGlobalTrailer . . . 122
5.16.4.20 kEnableJTAGReadout . . . 122
5.16.4.21 kEnableLocalHeader . . . 122
5.16.4.22 kEnableLocalTrailer . . . 123
5.16.4.23 kEnableMasterResetCode . . . 123
5.16.4.24 kEnableMasterResetOnEventReset . . . 123
5.16.4.25 kEnableMatching . . . 123
5.16.4.26 kEnableOverflowDetect . . . 123
5.16.4.27 kEnablePair . . . 123
5.16.4.28 kEnableReadoutOccupancy . . . 123
5.16.4.29 kEnableReadoutSeparator . . . 123
5.16.4.30 kEnableRelative . . . 123
5.16.4.31 kEnableResetChannelBufferWhenSeparator . . . 123
5.16.4.32 kEnableSeparatorOnBunchReset . . . 123
5.16.4.33 kEnableSeparatorOnEventReset . . . 123
5.16.4.34 kEnableSerial . . . 123
5.16.4.35 kEnableSetCountersOnBunchReset . . . 123
5.16.4.36 kEnableTTLClock . . . 123
5.16.4.37 kEnableTTLControl . . . 123
5.16.4.38 kEnableTTLHit . . . 123
5.16.4.39 kEnableTTLReset . . . 123
5.16.4.40 kEnableTTLSerial . . . 123
5.16.4.41 kEventCountOffset . . . 123
5.16.4.42 kIOClockDelay . . . 123
5.16.4.43 kIOClockSource . . . 123
5.16.4.44 kKeepToken . . . 123
5.16.4.45 kLeading . . . 123
5.16.4.46 kLeadingResolution . . . 123
5.16.4.47 kLowPowerMode . . . 123
5.16.4.48 kMaster . . . 123
5.16.4.49 kMatchWindow . . . 123

5.16.4.50 kMaxEventSize . . . 124
5.16.4.51 kModeRC . . . 124
5.16.4.52 kModeRCCompression . . . 124
5.16.4.53 kOffset0 . . . 124
5.16.4.54 kPLLControl . . . 124
5.16.4.55 kRCAdjust0 . . . 124
5.16.4.56 kReadoutFIFOSize . . . 124
5.16.4.57 kReadoutSingleCycleSpeed . . . 124
5.16.4.58 kReadoutSpeedSelect . . . 124
5.16.4.59 kRejectCountOffset . . . 124
5.16.4.60 kRejectFIFOFull . . . 124
5.16.4.61 kRollOver . . . 124
5.16.4.62 kSearchWindow . . . 124
5.16.4.63 kSelectBypassInputs . . . 124
5.16.4.64 kSerialClockDelay . . . 124
5.16.4.65 kSerialClockSource . . . 124
5.16.4.66 kSerialDelay . . . 124
5.16.4.67 kSetupParity . . . 124
5.16.4.68 kStrobeSelect . . . 124
5.16.4.69 kTDCId . . . 124
5.16.4.70 kTestInvert . . . 124
5.16.4.71 kTestMode . . . 124
5.16.4.72 kTestSelect . . . 124
5.16.4.73 kTokenDelay . . . 124
5.16.4.74 kTrailing . . . 124
5.16.4.75 kTriggerCountOffset . . . 124
5.16.4.76 kVernierOffset . . . 124
5.16.4.77 kWidthSelect . . . 125
5.17 TDCStatus Class Reference . . . 125
5.17.1 Detailed Description . . . 126
5.17.2 Constructor & Destructor Documentation . . . 126
5.17.2.1 TDCStatus . . . 126
5.17.2.2 TDCStatus . . . 126
5.17.3 Member Function Documentation . . . 126
5.17.3.1 SetConstantValues . . . 127
5.17.4 Field Documentation . . . 127
5.17.4.1 kDLLLock . . . 127
5.17.4.2 kError . . . 127
5.17.4.3 kHaveToken . . . 127
5.17.4.4 kL1Occupancy . . . 127

5.17.4.5 kReadoutFIFOEmpty . . . 127
5.17.4.6 kReadoutFIFOFull . . . 127
5.17.4.7 kReadoutFIFOOccupancy . . . 127
5.17.4.8 kTriggerFIFOEmpty . . . 127
5.17.4.9 kTriggerFIFOFull . . . 127
5.17.4.10 kTriggerFIFOOccupancy . . . 127
5.18 USBHandler Class Reference . . . 127
5.18.1 Detailed Description . . . 128
5.18.2 Constructor & Destructor Documentation . . . 128
5.18.2.1 USBHandler . . . 128
5.18.2.2 USBHandler . . . 128
5.18.2.3 ∼USBHandler . . . 128
5.18.3 Member Function Documentation . . . 128
5.18.3.1 DumpDevice . . . 128
5.18.3.2 Fetch . . . 128
5.18.3.3 Init . . . 129
5.18.3.4 Reset . . . 129
5.18.3.5 Write . . . 129
5.18.4 Field Documentation . . . 129
5.18.4.1 fDevice . . . 129
5.18.4.2 fHandle . . . 129
5.18.4.3 fIsStopping . . . 129
5.18.4.4 fProductId . . . 129
5.18.4.5 fVendorId . . . 129

Index **131**

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

FPGA board control 7
Socket communication objects 8
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
- FileReader . . . 23
- Message . . . 32
  - HTTPMessage . . . 29
  - SocketMessage . . . 47
- Socket . . . 41
  - Client . . . 15
    - FPGAHandler . . . 25
  - Messenger . . . 34
- TDC . . . 54
- TDCEvent . . . 68
- TDCRegister . . . 71
  - TDCBoundaryScan . . . 58
  - TDCControl . . . 61
  - TDCSetup . . . 75
  - TDCStatus . . . 125
- USBHandler . . . 127
  - FPGAHandler . . . 25

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Client
: Base client object for the socket . . . 15

Exception
: A simple exception handler . . . 20

file_header_t
: Header to the output files . . . 23

FileReader . . . 23

FPGAHandler
: Driver for timing detectors' FPGA readout . . . 25

HTTPMessage
: Message to be transmitted through a WebSocket protocol . . . 29

Message
: Base socket message type . . . 32

Messenger
: Base master object for the socket . . . 34

Socket
: Base socket object from which clients/master from a socket inherit . . . 41

SocketMessage
: Socket-passed message type . . . 47

TDC
: HPTDC object . . . 54

TDCBoundaryScan . . . 58

TDCControl
: Control word to be sent to the HPTDC chip . . . 61

TDCEvent
: HPTDC event parser . . . 68

TDCRegister
: General register object to interact with a HPTDC chip . . . 71

TDCSetup
: Setup word to be sent to the HPTDC chip . . . 75

TDCStatus . . . 125

USBHandler
: Generic USB communication handler . . . 127

Chapter 4

Module Documentation

4.1 FPGA board control

Data Structures

- class FPGAHandler

  Driver for timing detectors' FPGA readout.
- class USBHandler

  Generic USB communication handler.

4.1.1 Detailed Description

4.2 Socket communication objects

Data Structures

- class Client

  Base client object for the socket.
- class HTTPMessage

  Message to be transmitted through a WebSocket protocol.
- 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.

4.2.1 Detailed Description

4.2.2 Enumeration Type Documentation

4.2.2.1 enum Socket::SocketType

Type of actor playing a role on the socket.

Enumerator

INVALID

MASTER

WEBSOCKET_CLIENT

CLIENT

DETECTOR

4.3 HPTDC chip control

Data Structures

- class TDC

  HPTDC object.
- class TDCBoundaryScan
- class TDCControl

  Control word to be sent to the HPTDC chip.
- class TDCEvent

  HPTDC event parser.
- class TDCRegister

  General register object to interact with a HPTDC chip.
- class TDCSetup

  Setup word to be sent to the HPTDC chip.
- 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_80Mbits_s =0x1 }
- enum TDCSetup::SerialStrobeType { TDCSetup::SS_NoStrobe =0x0, TDCSetup::SS_DSStrobe =0x1, TDCSetup::SS_LeadingTrailingStrobe =0x2, TDCSetup::SS_LeadingEdge =0x3 }
- enum TDCSetup::ReadoutSingleCycleSpeed { TDCSetup::RSC_40Mbits_s =0x0, TDCSetup::RSC_20Mbits_s =0x1, TDCSetup::RSC_10Mbits_s =0x2, TDCSetup::RSC_5Mbits_s =0x3, TDCSetup::RSC_2p5Mbits_s =0x4, TDCSetup::RSC_1p25Mbits_s =0x5, TDCSetup::RSC_625kbits_s =0x6, TDCSetup::RSC_312p5kbits_s =0x7 }

4.3.1 Detailed Description

4.3.2 Enumeration Type Documentation

4.3.2.1 enum TDCSetup::CoreClockSource

Enumerator

Core_clock_40

Core_pll_clock_80

Core_pll_clock_160

Core_aux_clock

4.3.2.2 enum TDCSetup::DeadTime

Enumerator

DT_5ns

DT_10ns

DT_30ns

DT_100ns

4.3.2.3 enum TDCSetup::DLLClockSource

Enumerator

DLL_clock_40

DLL_pll_clock_40

DLL_pll_clock_160

DLL_pll_clock_320

DLL_aux_clock

4.3.2.4 enum TDCSetup::DLLSpeedMode

Enumerator

DLL_40MHz

DLL_160MHz

DLL_320MHz

DLL_Illegal

4.3.2.5 enum TDCSetup::EdgeResolution

Enumerator

E_100ps

E_200ps

E_400ps

E_800ps

E_1p6ns

E_3p12ns

E_6p25ns

E_12p5ns

4.3.2.6 enum TDCSetup::EnabledError

Enumerator

VernierError

CoarseError

ChannelSelectError

L1BufferParityError

TriggerFIFOParityError

TriggerMatchingError

ReadoutFIFOParityError

ReadoutStateError

SetupParityError

ControlParityError

JTAGInstructionParityError

4.3.2.7 enum TDCControl::EnablePattern

4.3.2.8 enum TDCEvent::EventType

Enumerator

Invalid

GroupHeader

GroupTrailer

TDCHeader

TDCTrailer

LeadingEdge

TrailingEdge

Error

Debug

4.3.2.9 enum TDCSetup::IOClockSource

Enumerator

IO_clock_40

IO_pll_clock_80

IO_pll_clock_160

IO_aux_clock

4.3.2.10 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

4.3.2.11 enum TDCSetup::ReadoutSpeed

Enumerator

RO_Fixed

RO_pll_80Mbits_s

4.3.2.12 enum TDCControl::RegisterName

Enumerator

R_EnablePattern

R_GlobalReset

R_DLLReset

R_PLLReset

4.3.2.13 enum TDCSetup::SerialClockSource

Enumerator

Serial_pll_clock_80

Serial_pll_clock_160

Serial_pll_clock_40

Serial_aux_clock

4.3.2.14 enum TDCSetup::SerialStrobeType

Enumerator

SS_NoStrobe

SS_DSStrobe

SS_LeadingTrailingStrobe

SS_LeadingEdge

4.3.2.15 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

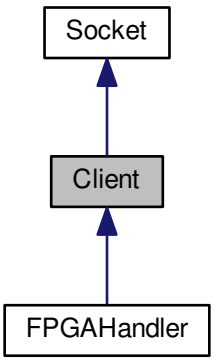
Data Structure Documentation

5.1 Client Class Reference

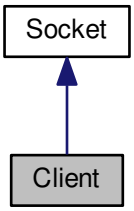
Base client object for the socket.

`#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.
- SocketMessage SendAndReceive (const SocketMessage &m, const MessageKey &a) const
- void Receive ()

  Receive a socket message from the master.
- virtual void ParseMessage (const SocketMessage &m)

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

  Socket actor type retrieval method.

Private Member Functions

- void Announce ()

  Announce our entry on the socket to its master.

Private Attributes

- int fClientId
- bool fIsConnected

Additional Inherited Members

5.1.1 Detailed Description

Base client object for the socket.

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

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

24 Mar 2015

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Client::Client ( ) [inline]

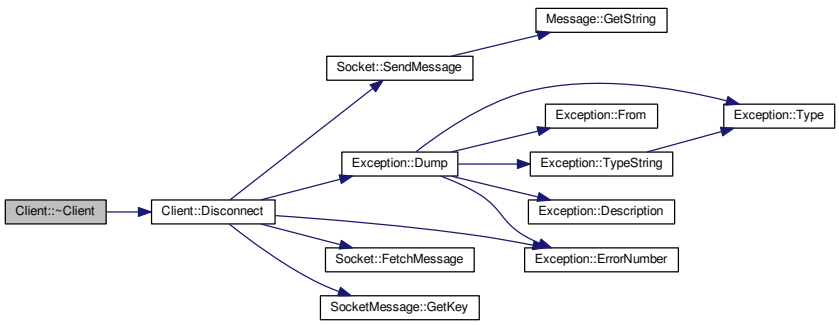
General void client constructor.

5.1.2.2 Client::Client (int *port*)

Bind a socket client to a given port.

5.1.2.3 Client::~Client ( ) [virtual]

Here is the call graph for this function:

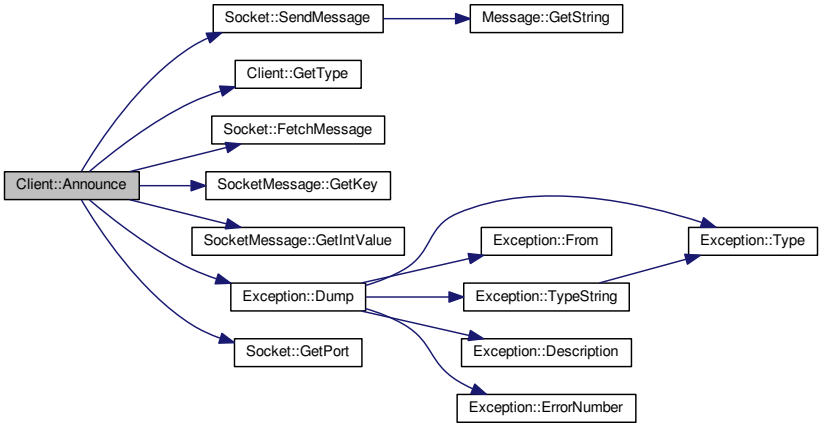


5.1.3 Member Function Documentation

5.1.3.1 void Client::Announce ( ) [private]

Announce our entry on the socket to its master.

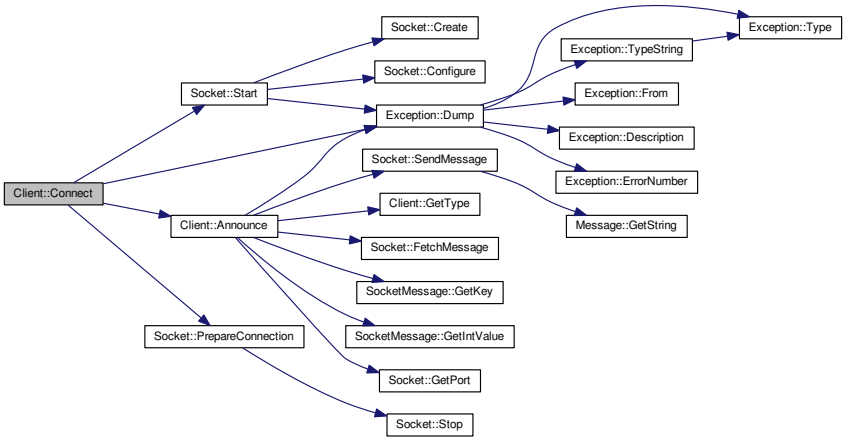
Here is the call graph for this function:



5.1.3.2 bool Client::Connect ( )

Bind this client to the socket.

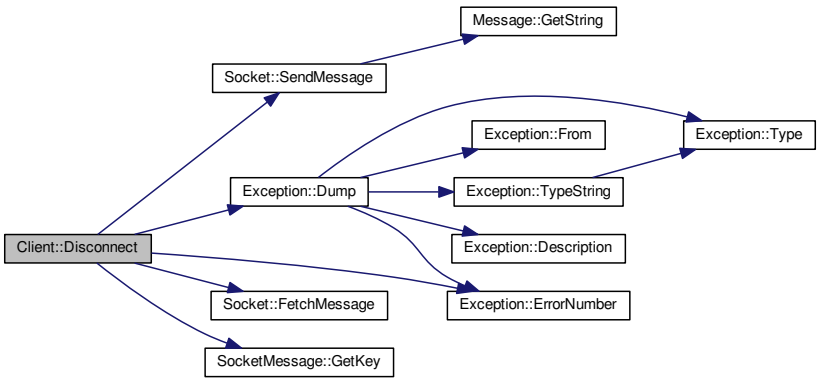
Here is the call graph for this function:



5.1.3.3 void Client::Disconnect ( )

Unbind this client from the socket.

Here is the call graph for this function:



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

Socket actor type retrieval method.

Reimplemented in FPGAHandler.

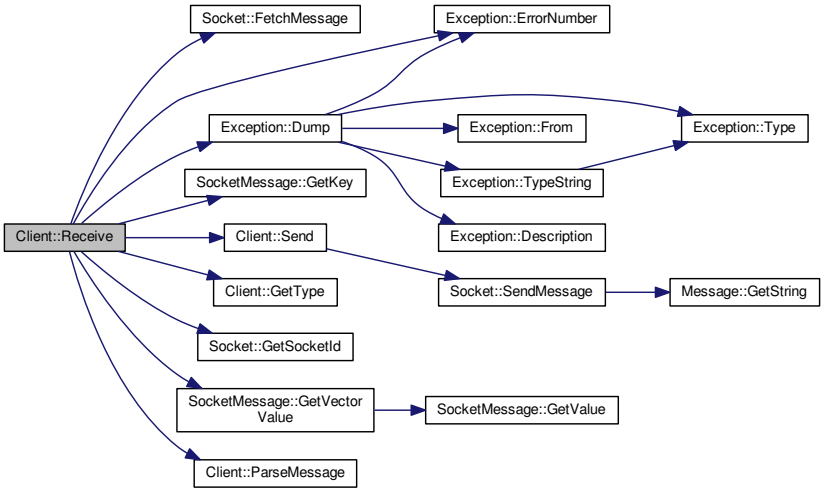
5.1.3.5 virtual void Client::ParseMessage (const SocketMessage & *m*) [inline], [virtual]

Parse a SocketMessage received from the master.

5.1.3.6 void Client::Receive ( )

Receive a socket message from the master.

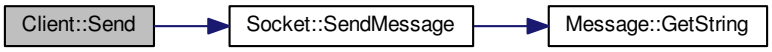
Here is the call graph for this function:



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

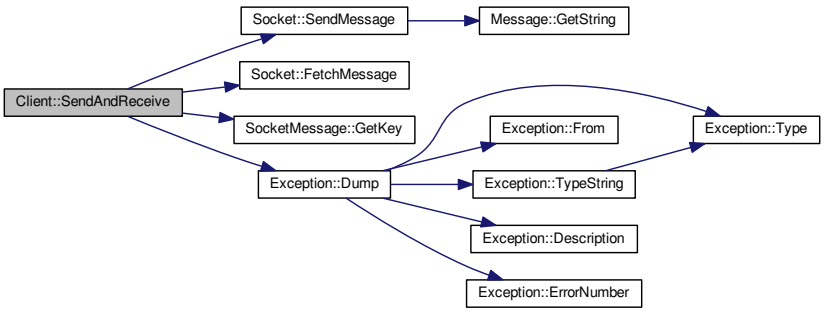
Send a message to the master through the socket.

Here is the call graph for this function:



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

Here is the call graph for this function:



5.1.4 Field Documentation

5.1.4.1 int Client::fClientId [private]

5.1.4.2 bool Client::fIsConnected [private]

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

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

5.2 Exception Class Reference

A simple exception handler.

`#include <Exception.h>`

Public Member Functions

- Exception (const char ∗from, std::string desc, ExceptionType type=Undefined, const int id=0)

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

Private Attributes

- std::string fFrom
- std::string fDescription
- ExceptionType fType
- int fErrorNumber

5.2.1 Detailed Description

A simple exception handler.

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

24 Mar 2015

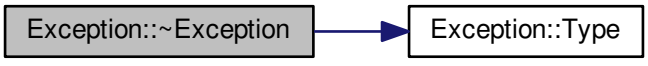
5.2.2 Constructor & Destructor Documentation

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

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

5.2.2.3 Exception::~Exception ( ) [inline]

Here is the call graph for this function:

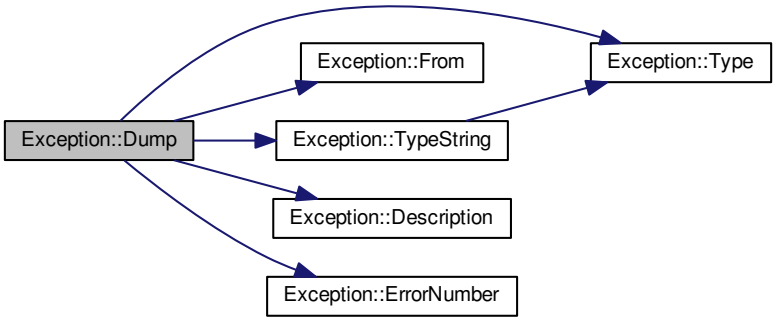


5.2.3 Member Function Documentation

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

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

Here is the call graph for this function:



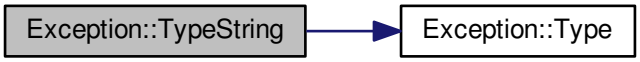
5.2.3.3 int Exception::ErrorNumber ( ) const [inline]

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

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

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

Here is the call graph for this function:



5.2.4 Field Documentation

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

5.2.4.2 int Exception::fErrorNumber [private]

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

5.2.4.4 ExceptionType Exception::fType [private]

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

- include/Exception.h

5.3 file_header_t Struct Reference

Header to the output files.

`#include <FileConstants.h>`

Data Fields

- uint32_t magic
- uint32_t run_id
- uint32_t spill_id
- uint8_t num_hptdc

5.3.1 Detailed Description

Header to the output files.

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

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

14 Apr 2015

5.3.2 Field Documentation

5.3.2.1 uint32_t file_header_t::magic

5.3.2.2 uint8_t file_header_t::num_hptdc

5.3.2.3 uint32_t file_header_t::run_id

5.3.2.4 uint32_t file_header_t::spill_id

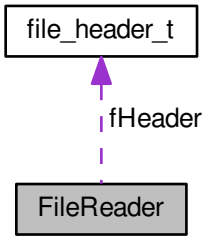
The documentation for this struct was generated from the following file:

- include/FileConstants.h

5.4 FileReader Class Reference

`#include <FileReader.h>`

Collaboration diagram for FileReader:



Public Member Functions

- FileReader (std::string name)
- ~FileReader ()
- unsigned int GetNumTDCs () const
- TDCEvent GetNextEvent ()

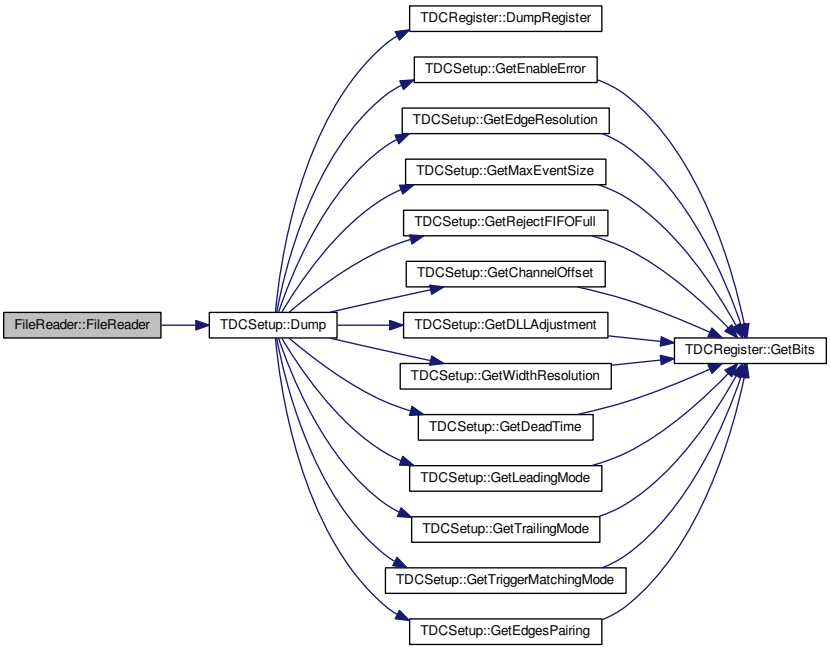
Private Attributes

- std::ifstream fFile
- file_header_t fHeader
- std::vector< TDCSetup > fSetupCollection

5.4.1 Constructor & Destructor Documentation

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

Here is the call graph for this function:



5.4.1.2 FileReader::~FileReader ( )

5.4.2 Member Function Documentation

5.4.2.1 TDCEvent FileReader::GetNextEvent ( ) [inline]

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

5.4.3 Field Documentation

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

5.4.3.2 file_header_t FileReader::fHeader [private]

5.4.3.3 std::vector<TDCSetup> FileReader::fSetupCollection [private]

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

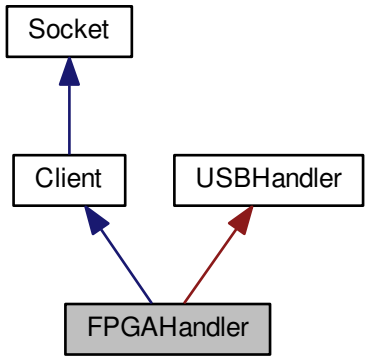
- include/FileReader.h
- src/FileReader.cpp

5.5 FPGAHandler Class Reference

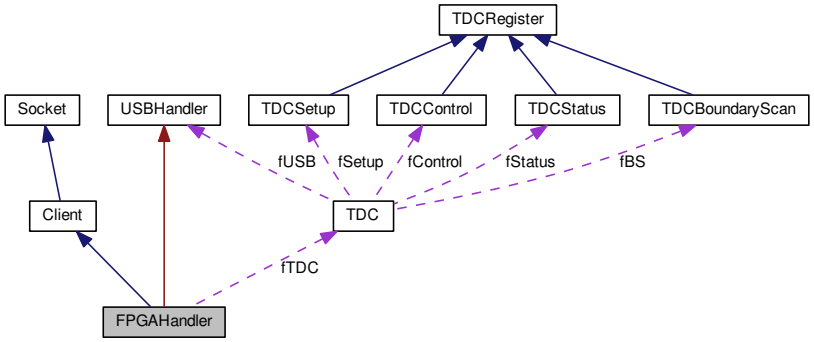
Driver for timing detectors' FPGA readout.

`#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.
- ~FPGAHandler ()
- void Stop ()
- void OpenFile ()

  Open an output file to store header/HPTDC events.
- void CloseFile ()

  Close a previously opened output file used to store header/HPTDC events.
- std::string GetFilename () const

  Retrieve the file name used to store data collected from the FPGA.
- TDC * GetTDC (unsigned int i=0)
- void SetTDCSetup (const TDCSetup &s)

- bool ErrorState ()
- int ReadBuffer ()
- SocketType GetType () const

  Socket actor type retrieval method.

Private Attributes

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

Additional Inherited Members

5.5.1 Detailed Description

Driver for timing detectors' FPGA readout.

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

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

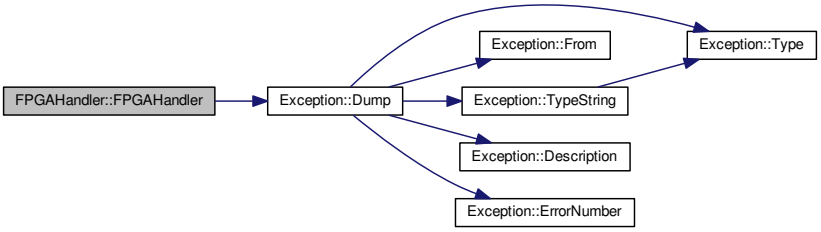
14 Apr 2015

5.5.2 Constructor & Destructor Documentation

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

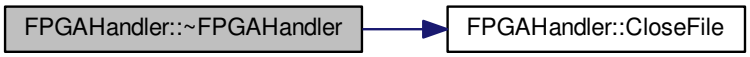
Bind to a FPGA through the USB protocol, and to the socket.

Here is the call graph for this function:



5.5.2.2 FPGAHandler::~FPGAHandler ( )

Here is the call graph for this function:



5.5.3 Member Function Documentation

5.5.3.1 void FPGAHandler::CloseFile ( )

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

5.5.3.2 bool FPGAHandler::ErrorState ( )

5.5.3.3 std::string FPGAHandler::GetFilename ( ) const [inline]

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

5.5.3.4 TDC* FPGAHandler::GetTDC (unsigned int *i* = 0) [inline]

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

Socket actor type retrieval method.

Reimplemented from Client.

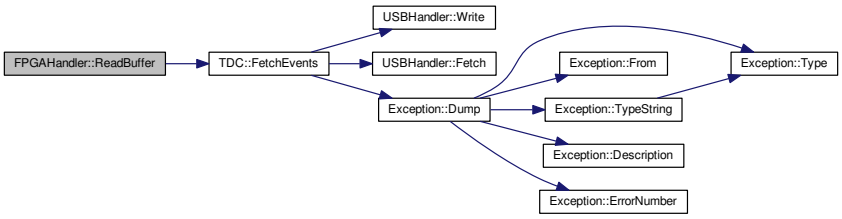
5.5.3.6 void FPGAHandler::OpenFile ( )

Open an output file to store header/HPTDC events.

Here is the call graph for this function:

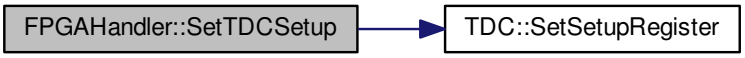
5.5.3.7 int FPGAHandler::ReadBuffer ( )

Here is the call graph for this function:



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

Here is the call graph for this function:



5.5.3.9 void FPGAHandler::Stop ( ) [inline]

5.5.4 Field Documentation

5.5.4.1 std::string FPGAHandler::fFilename [private]

5.5.4.2 bool FPGAHandler::fIsFileOpen [private]

5.5.4.3 bool FPGAHandler::fIsTDCInReadout [private]

5.5.4.4 std::ofstream FPGAHandler::fOutput [private]

5.5.4.5 TDC* FPGAHandler::fTDC[NUM_HPTDC] [private]

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

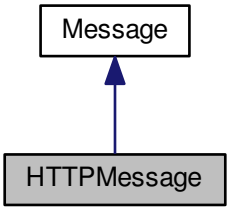
- include/FPGAHandler.h
- src/FPGAHandler.cpp

5.6 HTTPMessage Class Reference

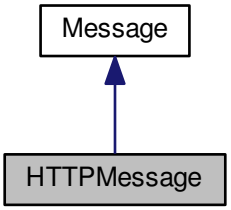
Message to be transmitted through a WebSocket protocol.

`#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

5.6.1 Detailed Description

Message to be transmitted through a WebSocket protocol.

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

Author

Laurent Forthomme laurent.forthomme@cern.ch

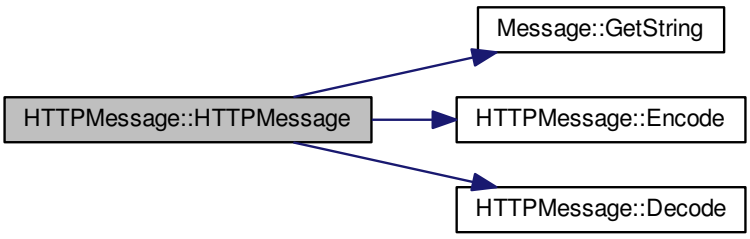
Date

1 Apr 2015

5.6.2 Constructor & Destructor Documentation

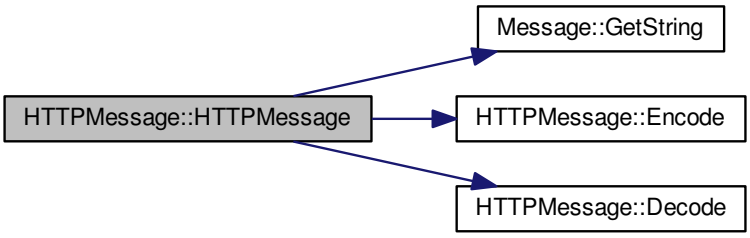
5.6.2.1 HTTPMessage::HTTPMessage (WebSocket * *ws*, Message *m*, MessageAction *a*) [inline]

Here is the call graph for this function:



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

Here is the call graph for this function:



5.6.3 Member Function Documentation

5.6.3.1 void HTTPMessage::Decode ( ) [inline]

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

5.6.3.3 void HTTPMessage::Encode ( ) [inline]

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

5.6.4 Field Documentation

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

5.6.4.2 WebSocket* HTTPMessage::fWS [private]

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

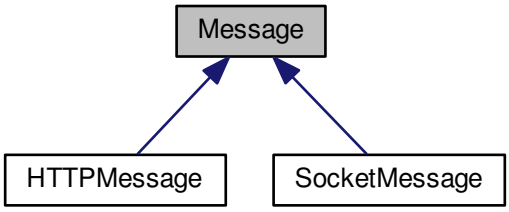
- include/HTTPMessage.h

5.7 Message Class Reference

Base socket message type.

`#include <Message.h>`

Inheritance diagram for Message:



Public Member Functions

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

Protected Attributes

- std::string fString

5.7.1 Detailed Description

Base socket message type.

Base handler for messages to be transmitted through the socket

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

6 Apr 2015

5.7.2 Constructor & Destructor Documentation

5.7.2.1 Message::Message ( ) [inline]

Void message constructor.

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

Construct a message from a string.

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

Construct a message from a string.

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

5.7.3 Member Function Documentation

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

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

Placeholder for the MessageKey retrieval method.

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

Retrieve the string carried by this message as a whole.

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

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

5.7.4 Field Documentation

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

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

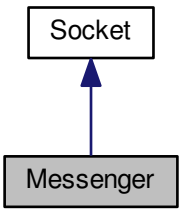
- include/Message.h

5.8 Messenger Class Reference

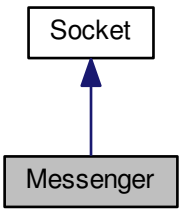
Base master object for the socket.

`#include <Messenger.h>`

Inheritance diagram for Messenger:



Collaboration diagram for Messenger:



Public Member Functions

- Messenger ()

  Build a void master object or socket actor.
- Messenger (int port)

  Build a master object to control the socket.
- ~Messenger ()
- bool Connect ()

  Connect the master to the socket.
- void Disconnect ()

  Remove the master and destroy the socket.
- void Send (const Message &m, int sid) const

  Send any type of message to any client.
- void Receive ()

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

Emit a message to all clients connected through the socket.

- void StartAcquisition ()

  Start the data acquisition.

- void StopAcquisition ()
- SocketType GetType () const

  Socket actor type retrieval method.

Private Member Functions

- void AddClient ()

  Add a client to listen to.

- void DisconnectClient (int sid, MessageKey key, bool force=false)

  Disconnect a client.

- void SwitchClientType (int sid, Socket::SocketType type)
- void ProcessMessage (SocketMessage m, int sid)

  Process a message received from the socket.

Private Attributes

- WebSocket * fWS
- int fNumAttempts

Additional Inherited Members

5.8.1 Detailed Description

Base master object for the socket.

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

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

23 Mar 2015

5.8.2 Constructor & Destructor Documentation

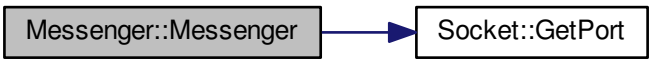
5.8.2.1 Messenger::Messenger ( )

Build a void master object or socket actor.

5.8.2.2 Messenger::Messenger (int *port*)

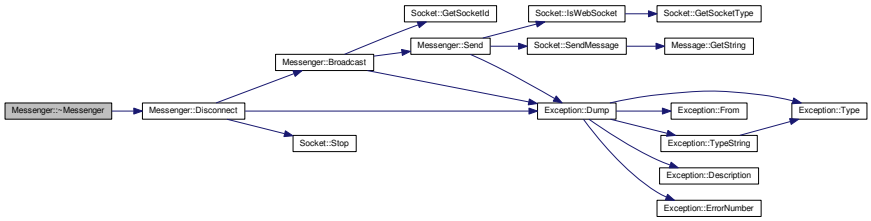
Build a master object to control the socket.

Here is the call graph for this function:



5.8.2.3 Messenger::~Messenger ( )

Here is the call graph for this function:



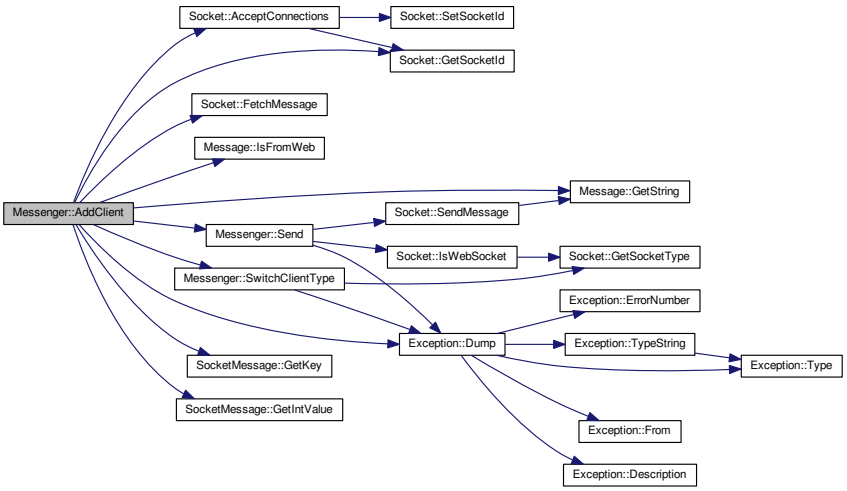
5.8.3 Member Function Documentation

5.8.3.1 void Messenger::AddClient ( ) [private]

Add a client to listen to.

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

Here is the call graph for this function:



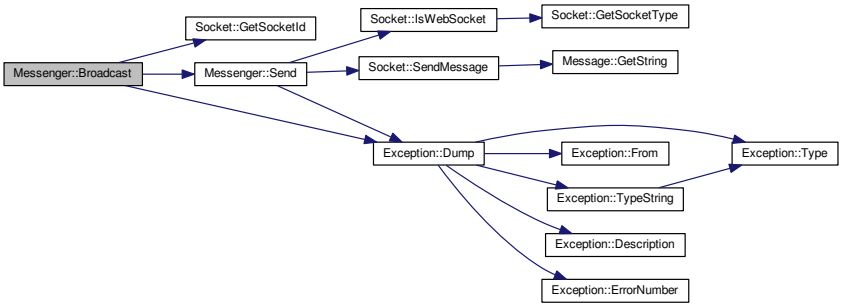
5.8.3.2 void Messenger::Broadcast (const Message & *m*) const

Emit a message to all clients connected through the socket.

Parameters

in	*m*	Message to transmit

Here is the call graph for this function:

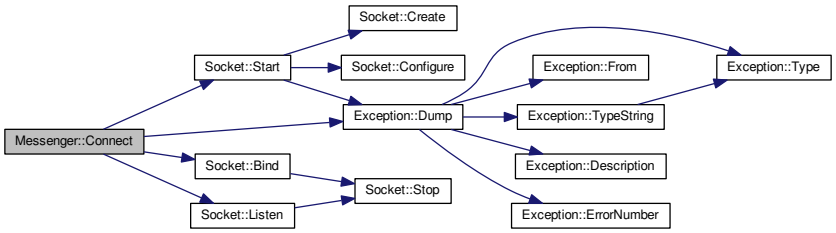


5.8.3.3 bool Messenger::Connect ( )

Connect the master to the socket.

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

Here is the call graph for this function:

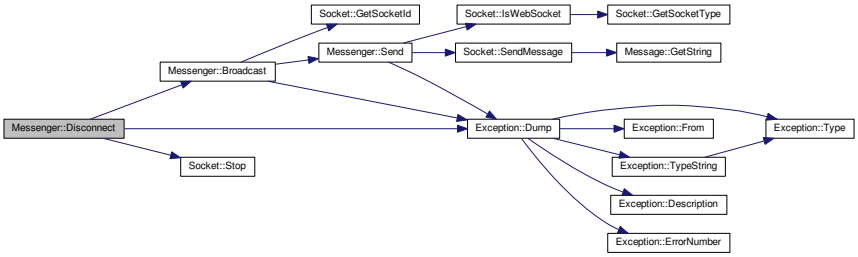


5.8.3.4 void Messenger::Disconnect ( )

Remove the master and destroy the socket.

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

Here is the call graph for this function:



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

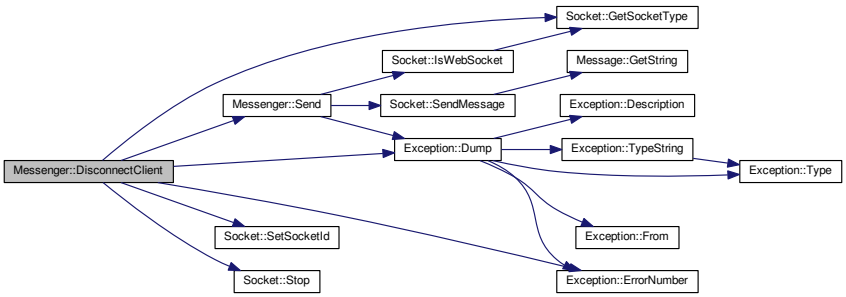
Disconnect a client.

Ask to a client to disconnect from this socket.

Parameters

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

Here is the call graph for this function:



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

Socket actor type retrieval method.

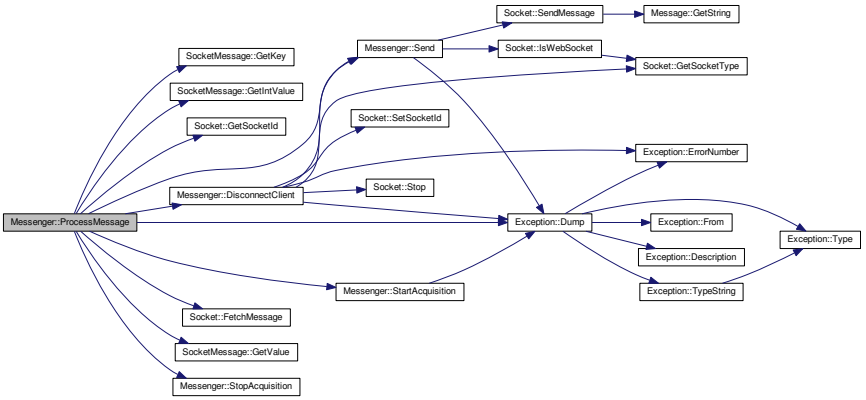
5.8.3.7 void Messenger::ProcessMessage (SocketMessage *m,* int *sid*) [private]

Process a message received from the socket.

Parameters

in	*Unique*	identifier of the client sending the message

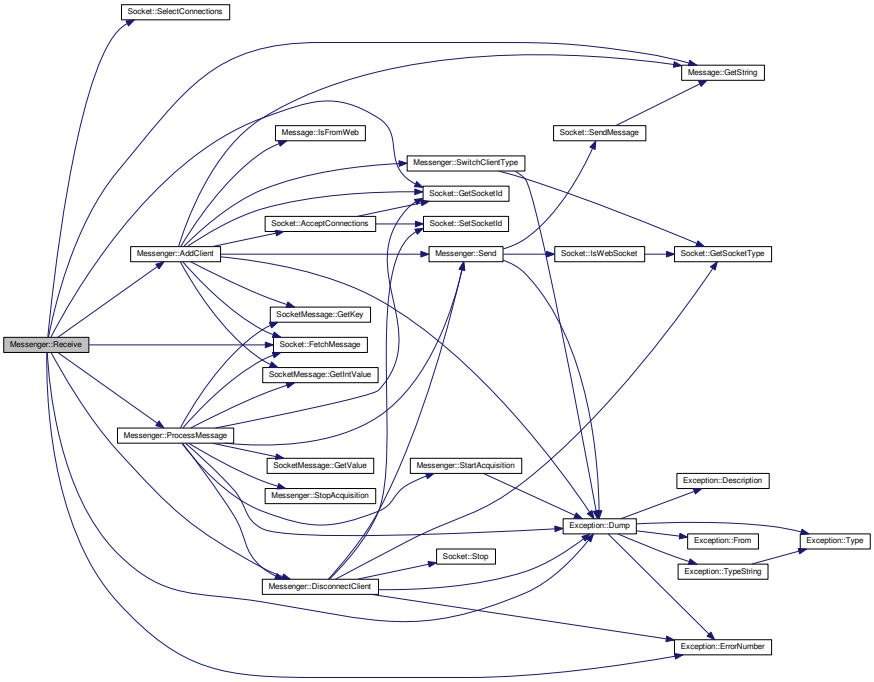
Here is the call graph for this function:



5.8.3.8 void Messenger::Receive ( )

Handle a message reception from a client.

Here is the call graph for this function:



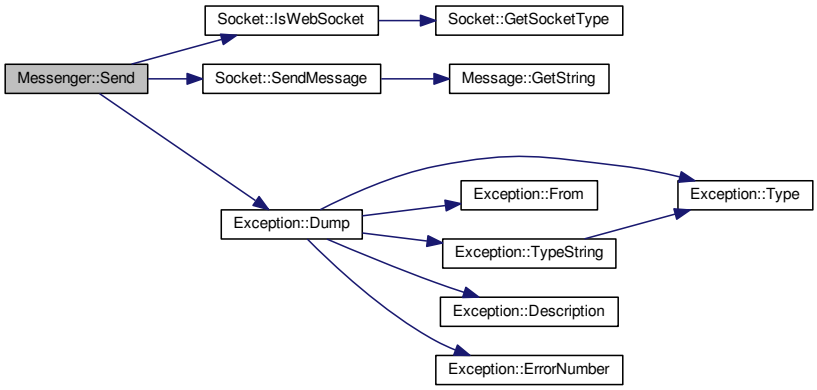
5.8.3.9 void Messenger::Send (const Message & *m,* int *sid*) const [inline]

Send any type of message to any client.

Parameters

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

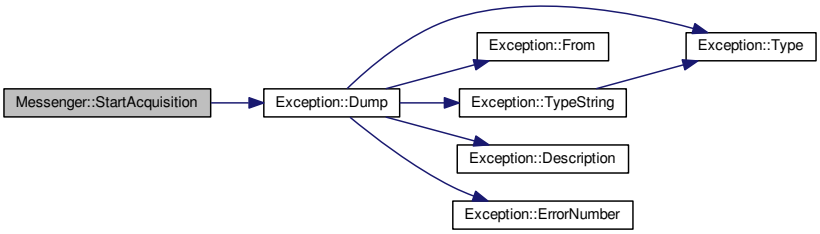
Here is the call graph for this function:



5.8.3.10 void Messenger::StartAcquisition ( )

Start the data acquisition.

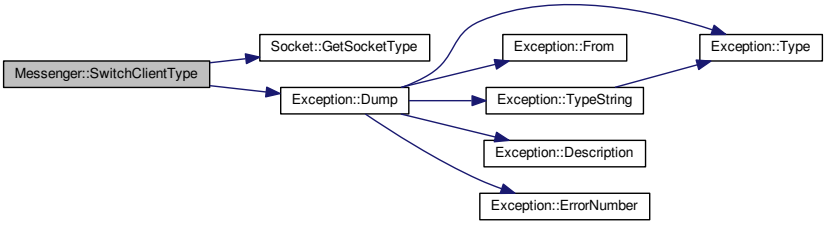
Here is the call graph for this function:



5.8.3.11 void Messenger::StopAcquisition ( )

5.8.3.12 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 int Messenger::fNumAttempts [private]

5.8.4.2 WebSocket* Messenger::fWS [private]

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

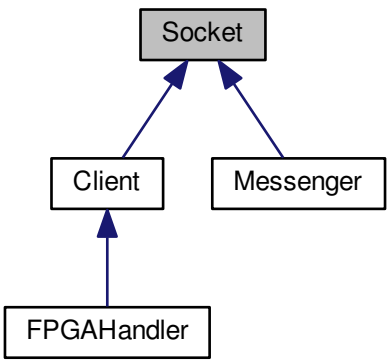
- include/Messenger.h
- src/Messenger.cpp

5.9 Socket Class Reference

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

`#include <Socket.h>`

Inheritance diagram for Socket:



Public Types

- enum SocketType {
  INVALID =-1, MASTER =0, WEBSOCKET_CLIENT, CLIENT,
  DETECTOR }

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

Public Member Functions

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

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

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

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

Protected Member Functions

- bool Start ()

  Start the socket.
- void Bind ()

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

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

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

  Receive a message from a socket.

Protected Attributes

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

  Master file descriptor list.
- fd_set fReadFds

  Temp file descriptor list for select()

Private Member Functions

- void Create ()

  Create an endpoint for communication.

- void Configure ()

  Configure the socket object for communication.

Private Attributes

- int fSocketId
- struct sockaddr_in fAddress

5.9.1 Detailed Description

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

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

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

23 Mar 2015

5.9.2 Member Typedef Documentation

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

5.9.3 Constructor & Destructor Documentation

5.9.3.1 Socket::Socket ( ) [inline]

5.9.3.2 Socket::Socket (int *port*)

5.9.3.3 Socket::~Socket ( ) [virtual]

5.9.4 Member Function Documentation

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

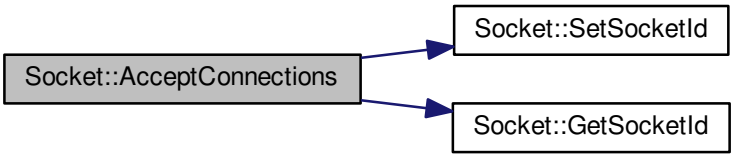
Accept connection from a client.

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

Parameters

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

Here is the call graph for this function:



5.9.4.2 void Socket::Bind ( ) [protected]

Bind a name to a socket.

Returns

Success of the operation

Here is the call graph for this function:

5.9.4.3 void Socket::Configure ( ) [private]

Configure the socket object for communication.

5.9.4.4 void Socket::Create ( ) [private]

Create an endpoint for communication.

5.9.4.5 void Socket::DumpConnected ( ) const

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

Receive a message from a socket.

Returns

Received message as a std::string

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

Retrieve the port used for this socket.

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

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

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

Here is the call graph for this function:

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

Listen to incoming messages.

Set the socket to listen to any message coming from outside

Here is the call graph for this function:

5.9.4.12 void Socket::PrepareConnection ( ) [protected]

Here is the call graph for this function:

5.9.4.13 void Socket::SelectConnections ( )

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

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

Send a message on a socket.

Here is the call graph for this function:

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

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

5.9.4.17 bool Socket::Start ( ) [protected]

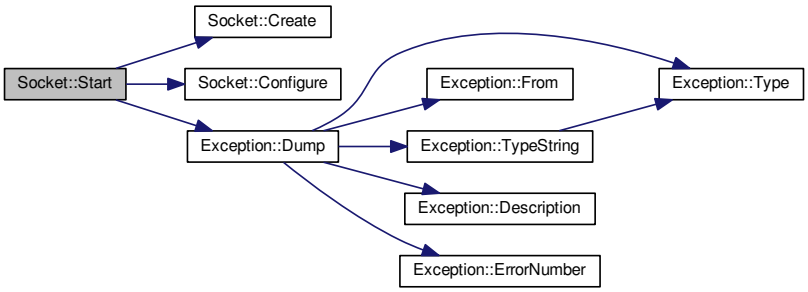
Start the socket.

Launch all mandatory operations to set the socket to be used

Returns

Success of the operation

Here is the call graph for this function:



5.9.4.18 void Socket::Stop ( )

Terminates the socket and all attached communications.

5.9.5 Field Documentation

5.9.5.1 struct sockaddr_in Socket::fAddress [private]

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

5.9.5.3 fd_set Socket::fMaster [protected]

Master file descriptor list.

5.9.5.4 int Socket::fPort [protected]

5.9.5.5 fd_set Socket::fReadFds [protected]

Temp file descriptor list for select()

5.9.5.6 int Socket::fSocketId [private]

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

5.9.5.7 SocketCollection Socket::fSocketsConnected [protected]

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

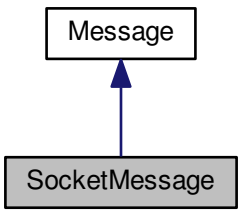
- include/Socket.h
- src/Socket.cpp

5.10 SocketMessage Class Reference

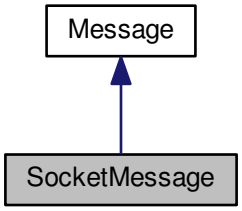
Socket-passed message type.

`#include <SocketMessage.h>`

Inheritance diagram for SocketMessage:



Collaboration diagram for SocketMessage:



Public Member Functions

- SocketMessage ()
- SocketMessage (const Message &msg)
- SocketMessage (const char *msg_s)
- SocketMessage (std::string msg_s)
- SocketMessage (const MessageKey &key)

  Construct a socket message out of a key.
- SocketMessage (const MessageKey &key, const char *value)

  Construct a socket message out of a key and a string-type value.
- SocketMessage (const MessageKey &key, std::string value)

  Construct a socket message out of a key and a string-type value.
- SocketMessage (const MessageKey &key, const int value)

  Construct a socket message out of a key and an integer-type value.
- SocketMessage (const MessageKey &key, const float value)

  Construct a socket message out of a key and a float-type value.
- SocketMessage (const MessageKey &key, const double value)

  Construct a socket message out of a key and a double precision-type value.
- SocketMessage (MessageMap msg_m)

  Construct a socket message out of a map of key/string-type value.
- ~SocketMessage ()
- void SetKeyValue (const MessageKey &key, const char *value)

  String-valued message.
- void SetKeyValue (const MessageKey &key, int int_value)

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

  Float-valued message.
- void SetKeyValue (const MessageKey &key, double double_value)

  Double-valued message.
- std::string GetString () const

  Extract the whole key:value message.
- MessageKey GetKey () const

  Extract the message's key.
- std::string GetValue () const

  Extract the message's string value.
- int GetIntValue () const

Extract the message's integer value.

- VectorValue GetVectorValue () const

  Extract the message's vector of string value.

- void Dump (std::ostream &os=std::cout) const

Private Member Functions

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

Private Attributes

- MessageMap fMessage

Additional Inherited Members

5.10.1 Detailed Description

Socket-passed message type.

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

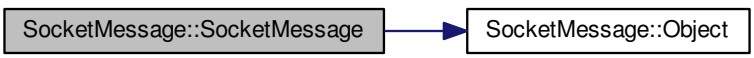
26 Mar 2015

5.10.2 Constructor & Destructor Documentation

5.10.2.1 SocketMessage::SocketMessage ( ) [inline]

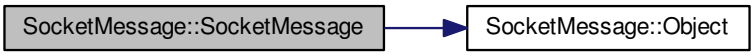
5.10.2.2 SocketMessage::SocketMessage (const Message & *msg*) [inline]

Here is the call graph for this function:



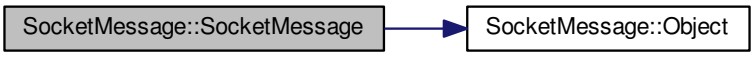
5.10.2.3 SocketMessage::SocketMessage (const char * *msg_s*) [inline]

Here is the call graph for this function:



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

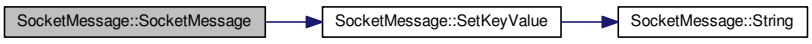
Here is the call graph for this function:



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

Construct a socket message out of a key.

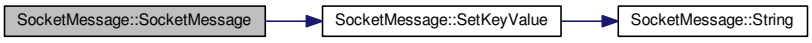
Here is the call graph for this function:



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

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

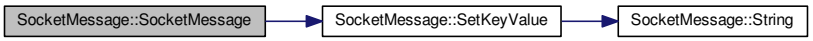
Here is the call graph for this function:



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

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

Here is the call graph for this function:



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

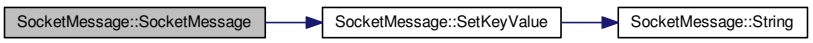
Construct a socket message out of a key and an integer-type value.

Here is the call graph for this function:

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

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

Here is the call graph for this function:



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

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

Here is the call graph for this function:

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

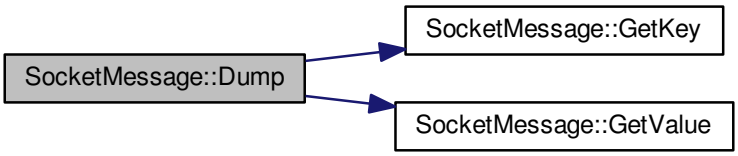
Construct a socket message out of a map of key/string-type value.

5.10.2.12 SocketMessage::~SocketMessage ( ) [inline]

5.10.3 Member Function Documentation

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

Here is the call graph for this function:



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

Extract the message's integer value.

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

Extract the message's key.

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

Extract the whole key:value message.

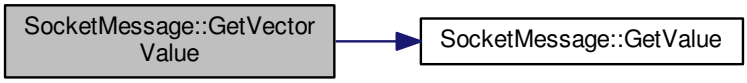
5.10.3.5 std::string SocketMessage::GetValue ( ) const [inline]

Extract the message's string value.

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

Extract the message's vector of string value.

Here is the call graph for this function:

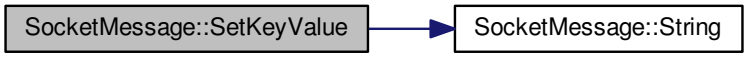


5.10.3.7 MessageMap SocketMessage::Object ( ) const [inline], [private]

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

String-valued message.

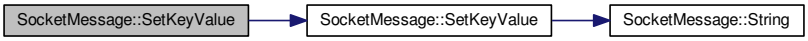
Here is the call graph for this function:



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

Send an integer-valued message.

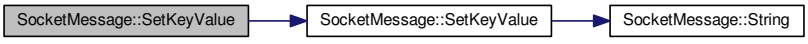
Here is the call graph for this function:



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

Float-valued message.

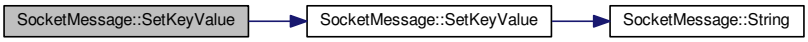
Here is the call graph for this function:



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

Double-valued message.

Here is the call graph for this function:



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

5.10.4 Field Documentation

5.10.4.1 MessageMap SocketMessage::fMessage [private]

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

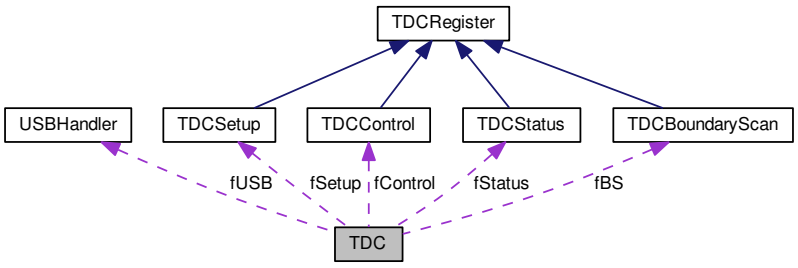
- include/SocketMessage.h

5.11 TDC Class Reference

HPTDC object.

`#include <TDC.h>`

Collaboration diagram for TDC:



Public Member Functions

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

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

  Retrieve the HPTDC setup word as a TDCSetup object.
- bool CheckFirmwareVersion () const

- void SoftReset ()
- TDCEventCollection FetchEvents ()
- void ReadStatus ()

Private Member Functions

- void SendConfiguration ()

  Set the setup word to the HPTDC internal setup register.
- void ReadConfiguration ()

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

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

  Retrieve one register content from the HPTDC inner memory.

Private Attributes

- unsigned int fId
- USBHandler * fUSB
- TDCSetup fSetup
- TDCControl fControl
- TDCBoundaryScan fBS
- TDCStatus fStatus

5.11.1 Detailed Description

HPTDC object.

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

27 Apr 2015

5.11.2 Constructor & Destructor Documentation

5.11.2.1 TDC::TDC (unsigned int *id*, USBHandler * *h*)

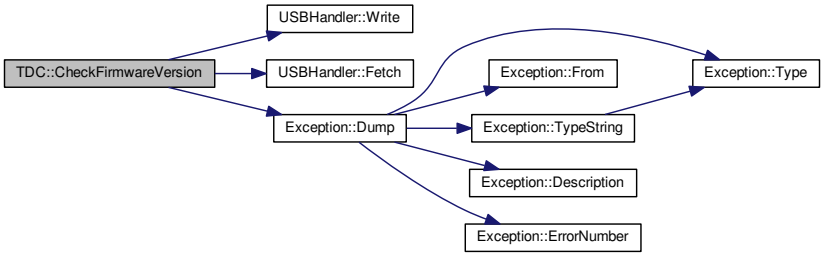
Here is the call graph for this function:

5.11.2.2 TDC::~TDC ( ) [inline]

5.11.3 Member Function Documentation

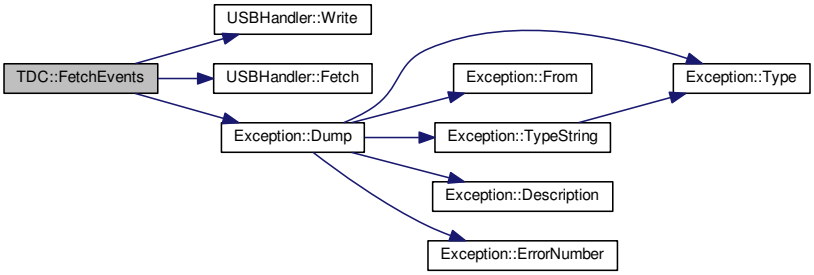
5.11.3.1 bool TDC::CheckFirmwareVersion ( ) const

Here is the call graph for this function:



5.11.3.2 TDCEventCollection TDC::FetchEvents ( )

Here is the call graph for this function:



5.11.3.3 TDCSetup TDC::GetSetupRegister ( ) [inline]

Retrieve the HPTDC setup word as a TDCSetup object.

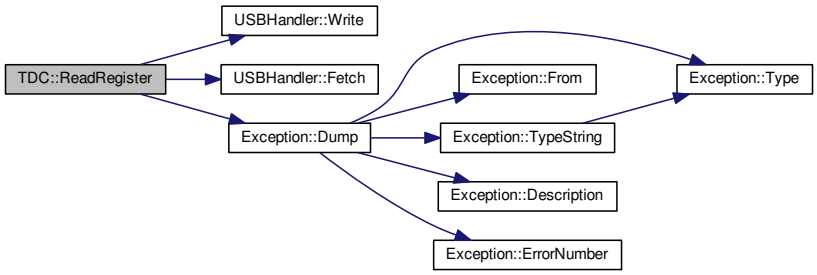
5.11.3.4 void TDC::ReadConfiguration ( ) [private]

Read the setup word from the HPTDC internal setup register.

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

Retrieve one register content from the HPTDC inner memory.

Here is the call graph for this function:



5.11.3.6 void TDC::ReadStatus ( ) [inline]

5.11.3.7 void TDC::SendConfiguration ( ) [private]

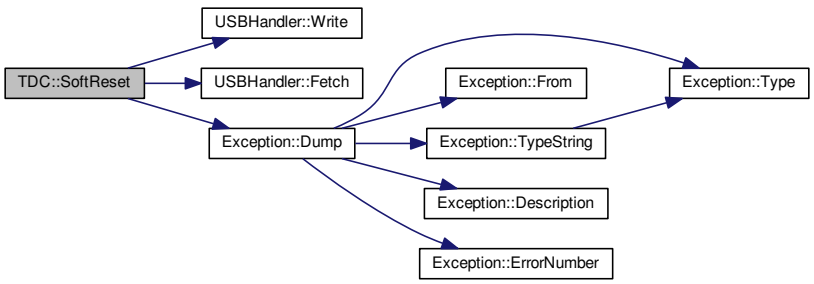
Set the setup word to the HPTDC internal setup register.

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

Submit the HPTDC setup word as a TDCSetup object.

5.11.3.9 void TDC::SoftReset ( )

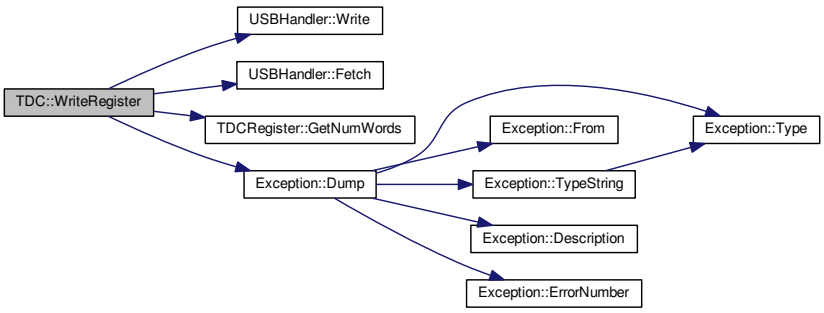
Here is the call graph for this function:



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

Write one register content on the HPTDC inner memory.

Here is the call graph for this function:



5.11.4 Field Documentation

5.11.4.1 TDCBoundaryScan TDC::fBS [private]

5.11.4.2 TDCControl TDC::fControl [private]

5.11.4.3 unsigned int TDC::fId [private]

5.11.4.4 TDCSetup TDC::fSetup [private]

5.11.4.5 TDCStatus TDC::fStatus [private]

5.11.4.6 USBHandler* TDC::fUSB [private]

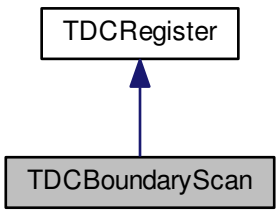
The documentation for this class was generated from the following files:

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

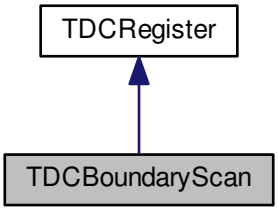
5.12 TDCBoundaryScan Class Reference

`#include <TDCBoundaryScan.h>`

Inheritance diagram for TDCBoundaryScan:



Collaboration diagram for TDCBoundaryScan:



Public Member Functions

- TDCBoundaryScan ()
- TDCBoundaryScan (const TDCBoundaryScan &bs)
- void SetConstantValues ()

Static Private Attributes

- static const bit kTokenOut = 0
- static const bit kStrobeOut = 1
- static const bit kSerialOut = 2
- static const bit kTest = 3
- static const bit kError = 4
- static const bit kDataReady = 5
- static const bit kParallelEnable = 6
- static const bit kParallelDataOut = 7
- static const bit kEncodedControl = 39
- static const bit kTrigger = 40
- static const bit kEventReset = 41
- static const bit kBunchReset = 42
- static const bit kGetData = 43
- static const bit kSerialBypassIn = 44
- static const bit kSerialIn = 45
- static const bit kTokenBypassIn = 46
- static const bit kTokenIn = 47
- static const bit kReset = 48
- static const bit kAuxClock = 49
- static const bit kClk = 50
- static const bit kHit = 51

Additional Inherited Members

5.12.1 Detailed Description

Author

Laurent Forthomme laurent.forthomme@cern.ch

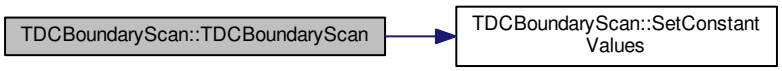
Date

24 Apr 2015

5.12.2 Constructor & Destructor Documentation

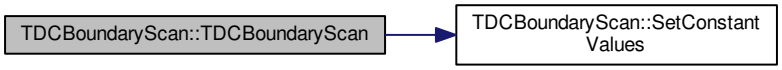
5.12.2.1 TDCBoundaryScan::TDCBoundaryScan ( ) [inline]

Here is the call graph for this function:



5.12.2.2 TDCBoundaryScan::TDCBoundaryScan (const TDCBoundaryScan & *bs*) [inline]

Here is the call graph for this function:



5.12.3 Member Function Documentation

5.12.3.1 void TDCBoundaryScan::SetConstantValues ( ) [inline], [virtual]

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

Implements TDCRegister.

5.12.4 Field Documentation

5.12.4.1 const bit TDCBoundaryScan::kAuxClock = 49 [static], [private]

5.12.4.2 const bit TDCBoundaryScan::kBunchReset = 42 [static], [private]

5.12.4.3 const bit TDCBoundaryScan::kClk = 50 [static], [private]

5.12.4.4 const bit TDCBoundaryScan::kDataReady = 5 [static], [private]

5.12.4.5 const bit TDCBoundaryScan::kEncodedControl = 39 [static], [private]

5.12.4.6 const bit TDCBoundaryScan::kError = 4 [static], [private]

5.12.4.7 const bit TDCBoundaryScan::kEventReset = 41 [static], [private]

5.12.4.8 const bit TDCBoundaryScan::kGetData = 43 [static], [private]

5.12.4.9 const bit TDCBoundaryScan::kHit = 51 [static], [private]

5.12.4.10 const bit TDCBoundaryScan::kParallelDataOut = 7 [static], [private]

5.12.4.11 const bit TDCBoundaryScan::kParallelEnable = 6 [static], [private]

5.12.4.12 const bit TDCBoundaryScan::kReset = 48 [static], [private]

5.12.4.13 const bit TDCBoundaryScan::kSerialBypassIn = 44 [static], [private]

5.12.4.14 const bit TDCBoundaryScan::kSerialIn = 45 [static], [private]

5.12.4.15 const bit TDCBoundaryScan::kSerialOut = 2 [static], [private]

5.12.4.16 const bit TDCBoundaryScan::kStrobeOut = 1 [static], [private]

5.12.4.17 const bit TDCBoundaryScan::kTest = 3 [static], [private]

5.12.4.18 const bit TDCBoundaryScan::kTokenBypassIn = 46 [static], [private]

5.12.4.19 const bit TDCBoundaryScan::kTokenIn = 47 [static], [private]

5.12.4.20 const bit TDCBoundaryScan::kTokenOut = 0 [static], [private]

5.12.4.21 const bit TDCBoundaryScan::kTrigger = 40 [static], [private]

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

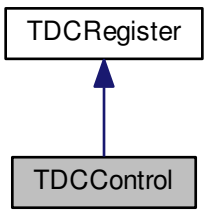
- include/TDCBoundaryScan.h

5.13 TDCControl Class Reference

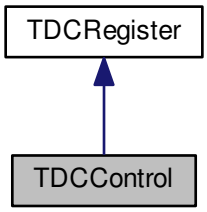
Control word to be sent to the HPTDC chip.

`#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

5.13.1 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

Date

24 Apr 2015

5.13.2 Constructor & Destructor Documentation

5.13.2.1 TDCControl::TDCControl ( ) [inline]

Here is the call graph for this function:

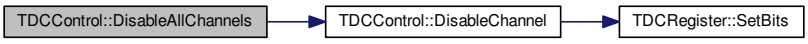
5.13.2.2 TDCControl::TDCControl (const TDCControl & *c*) [inline]

Here is the call graph for this function:

5.13.3 Member Function Documentation

5.13.3.1 void TDCControl::DisableAllChannels ( ) [inline]

Here is the call graph for this function:



5.13.3.2 void TDCControl::DisableChannel (unsigned int *id*) [inline]

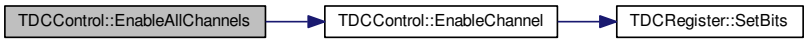
Here is the call graph for this function:

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

Here is the call graph for this function:

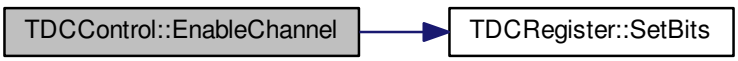
5.13.3.4 void TDCControl::EnableAllChannels ( ) [inline]

Here is the call graph for this function:



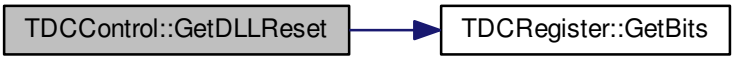
5.13.3.5 void TDCControl::EnableChannel (unsigned int *id*) [inline]

Here is the call graph for this function:



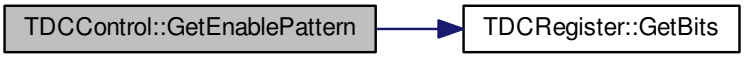
5.13.3.6 bool TDCControl::GetDLLReset ( ) const [inline]

Here is the call graph for this function:



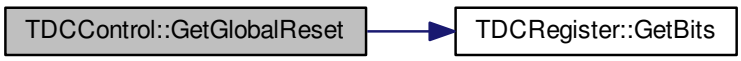
5.13.3.7 EnablePattern TDCControl::GetEnablePattern ( ) const [inline]

Here is the call graph for this function:



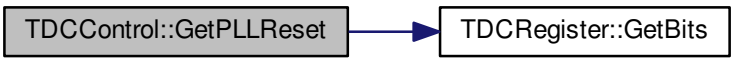
5.13.3.8 bool TDCControl::GetGlobalReset ( ) const [inline]

Here is the call graph for this function:



5.13.3.9 bool TDCControl::GetPLLReset ( ) const [inline]

Here is the call graph for this function:



5.13.3.10 void TDCControl::SetConstantValues ( ) [inline], [virtual]

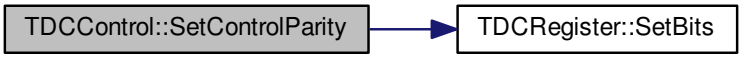
Ensure that the critical constant values are properly set in the register word

Implements TDCRegister.

Here is the call graph for this function:

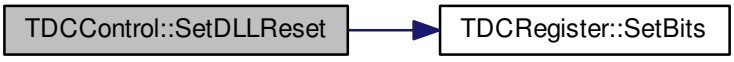
5.13.3.11 void TDCControl::SetControlParity (const bool *cp* = true) [inline], [private]

Here is the call graph for this function:



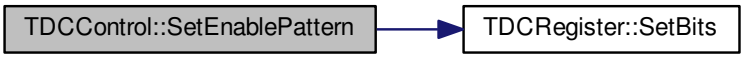
5.13.3.12 void TDCControl::SetDLLReset (const bool *dr* = true) [inline]

Here is the call graph for this function:



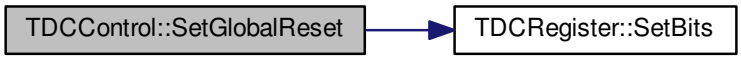
5.13.3.13 void TDCControl::SetEnablePattern (const EnablePattern & *ep*) [inline]

Here is the call graph for this function:



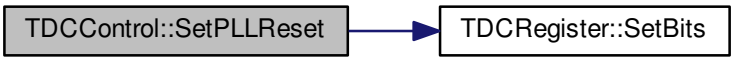
5.13.3.14 void TDCControl::SetGlobalReset (const bool *gr* = true) [inline]

Here is the call graph for this function:



5.13.3.15 void TDCControl::SetPLLReset (const bool *pr* = true) [inline]

Here is the call graph for this function:



5.13.4 Field Documentation

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

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

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

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

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

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

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

- include/TDCControl.h

5.14 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 ()
- TDCEvent (const uint32_t &word)
- virtual ~TDCEvent ()
- void SetWord (const uint32_t &word)
- 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

5.14.1 Detailed Description

HPTDC event parser.

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

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

20 Apr 2015

5.14.2 Constructor & Destructor Documentation

5.14.2.1 TDCEvent::TDCEvent ( ) [inline]

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

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

5.14.3 Member Function Documentation

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

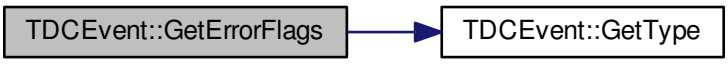
Bunch identifier of trigger (or trigger time tag)

Here is the call graph for this function:

5.14.3.2 uint16_t TDCEvent::GetErrorFlags ( ) const [inline]

Return error flags if an error condition has been detected.

Here is the call graph for this function:



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

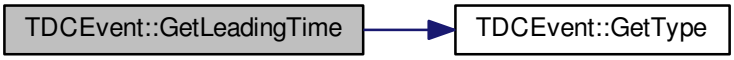
Event identifier from event counter.

Here is the call graph for this function:

5.14.3.4 uint32_t TDCEvent::GetLeadingTime (bool *pair* = false) const [inline]

Leading edge measurement in programmed time resolution.

Here is the call graph for this function:



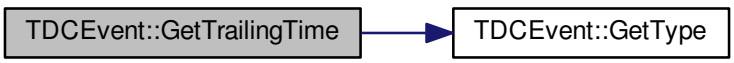
5.14.3.5 unsigned int TDCEvent::GetTDCId ( ) const [inline]

Programmed identifier of master TDC.

5.14.3.6 uint32_t TDCEvent::GetTrailingTime ( ) const [inline]

Trailing edge measurement in programmed time resolution.

Here is the call graph for this function:



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

Type of packet read out from the TDC.

5.14.3.8 uint8_t TDCEvent::GetWidth ( ) const [inline]

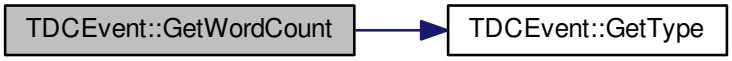
Width of pulse in programmed time resolution.

Here is the call graph for this function:

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

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

Here is the call graph for this function:



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

5.14.4 Field Documentation

5.14.4.1 uint32_t TDCEvent::fWord [private]

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

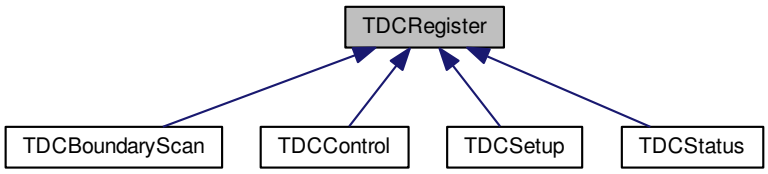
- include/TDCEvent.h

5.15 TDCRegister Class Reference

General register object to interact with a HPTDC chip.

`#include <TDCRegister.h>`

Inheritance diagram for TDCRegister:



Public Types

- typedef uint16_t bit

  LSB index.
- typedef uint32_t word_t

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

Public Member Functions

- TDCRegister (const unsigned int size)
- TDCRegister (const unsigned int size, const TDCRegister &r)
- virtual ~TDCRegister ()
- TDCRegister & operator= (const TDCRegister &r)
- 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.

5.15.1 Detailed Description

General register object to interact with a HPTDC chip.

Author

Laurent Forthomme laurent.forthomme@cern.ch

Date

24 Apr 2015

5.15.2 Member Typedef Documentation

5.15.2.1 typedef uint16_t TDCRegister::bit

LSB index.

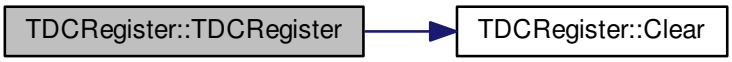
5.15.2.2 typedef uint32_t TDCRegister::word_t

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

5.15.3 Constructor & Destructor Documentation

5.15.3.1 TDCRegister::TDCRegister (const unsigned int *size*) [inline]

Here is the call graph for this function:



5.15.3.2 TDCRegister::TDCRegister (const unsigned int *size*, const TDCRegister & *r*) [inline]

Here is the call graph for this function:

5.15.3.3 virtual TDCRegister::~TDCRegister ( ) [inline], [virtual]

5.15.4 Member Function Documentation

5.15.4.1 void TDCRegister::Clear ( ) [inline], [protected]

Set all bits in this register to '0'.

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

5.15.4.3 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

in	*lsb*	Least significant bit of the word to retrieve
in	*size*	Size of the word to retrieve

5.15.4.4 uint8_t TDCRegister::GetNumWords ( ) const [inline]

Number of words in the register.

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

5.15.4.5 word_t TDCRegister::GetWord (const unsigned int *i*) const [inline]

Retrieve one subset from the register word.

5.15.4.6 TDCRegister& TDCRegister::operator= (const TDCRegister & *r*) [inline]

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

Set bits in the register word.

Set a fixed amount of bits in the full register word

Parameters

in	*lsb*	Least significant bit of the word to set
in	*word*	Word to set
in	*size*	Size of the word to set

5.15.4.8 virtual void TDCRegister::SetConstantValues ( ) [pure virtual]

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

Implemented in TDCSetup, TDCControl, TDCBoundaryScan, and TDCStatus.

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

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

5.15.5 Field Documentation

5.15.5.1 unsigned int TDCRegister::fNumWords [protected]

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

5.15.5.2 word_t* TDCRegister::fWord [protected]

Pointer to this register's word.

5.15.5.3 unsigned int TDCRegister::fWordSize [protected]

Number of bits in this register.

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

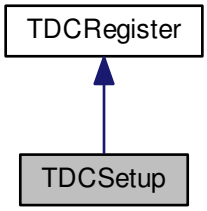
- include/TDCRegister.h

5.16 TDCSetup Class Reference

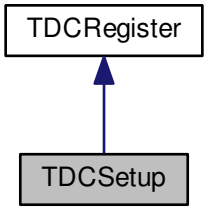
Setup word to be sent to the HPTDC chip.

`#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, ReadoutState↩
  Error =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_DSStrobe =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=0xFFF)

  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

5.16.1 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

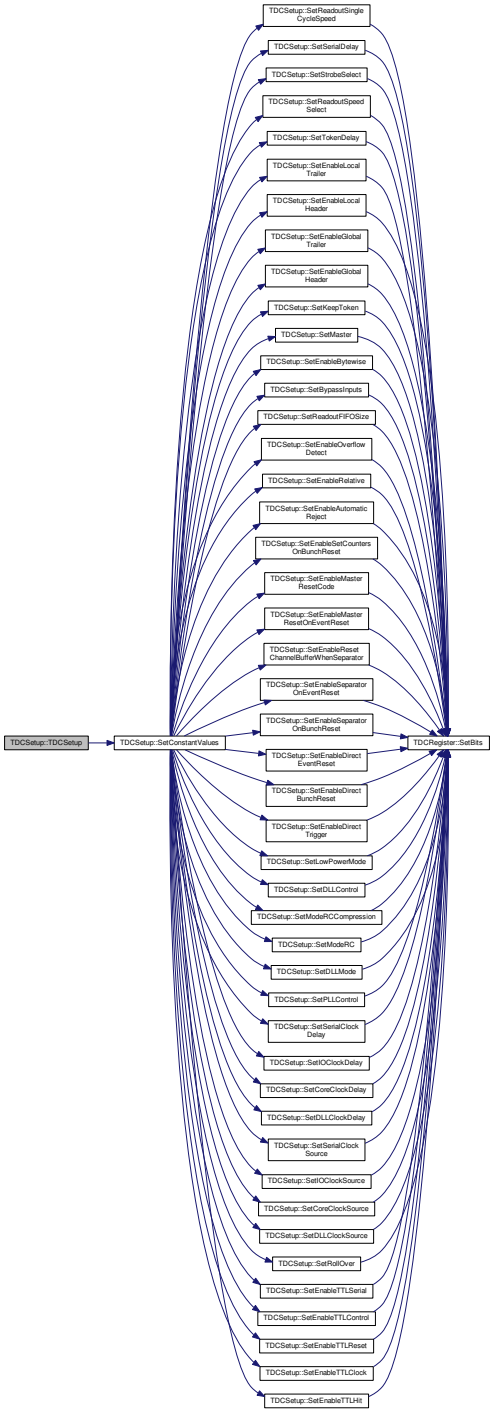
Date

16 Apr 2015

5.16.2 Constructor & Destructor Documentation

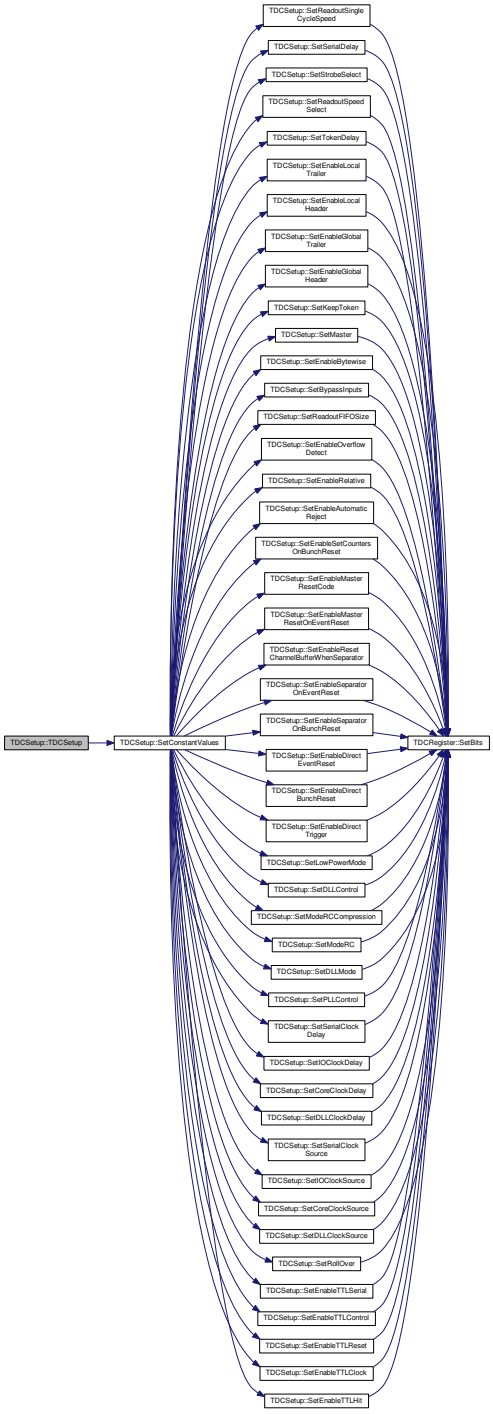
5.16.2.1 TDCSetup::TDCSetup ( ) [inline]

Here is the call graph for this function:



5.16.2.2 TDCSetup::TDCSetup (const TDCSetup & *c*) [inline]

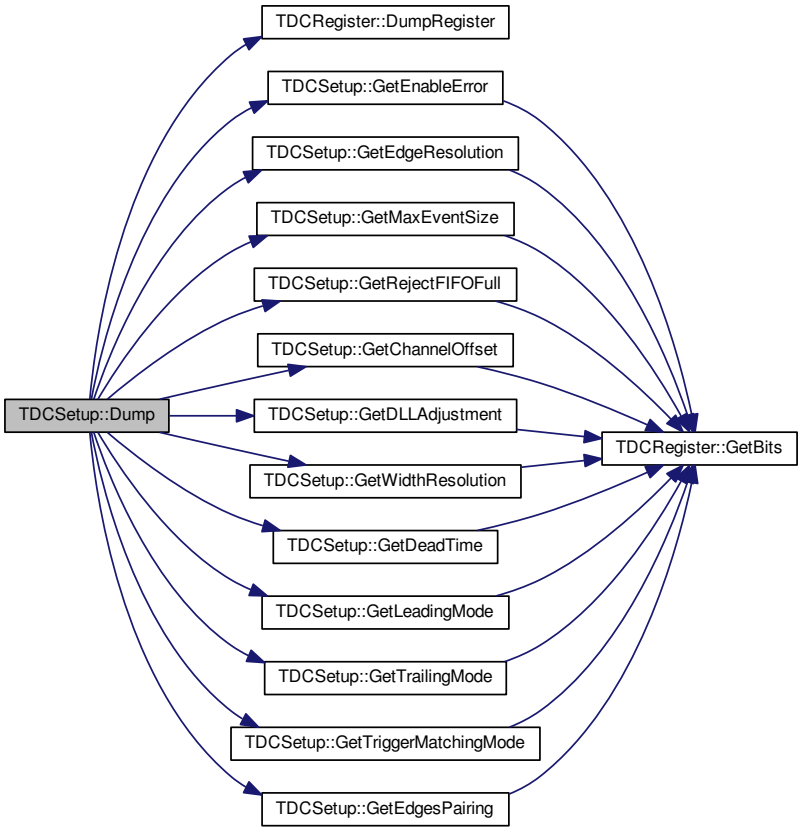
Here is the call graph for this function:



5.16.3 Member Function Documentation

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

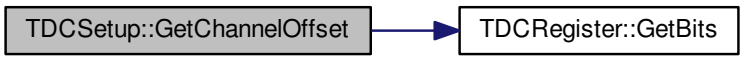
Here is the call graph for this function:



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

Return the offset for one single channel.

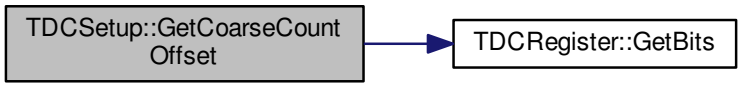
Here is the call graph for this function:



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

Extract offset for the coarse time counter.

Here is the call graph for this function:



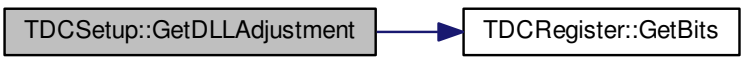
5.16.3.4 DeadTime TDCSetup::GetDeadTime ( ) const [inline]

Here is the call graph for this function:

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

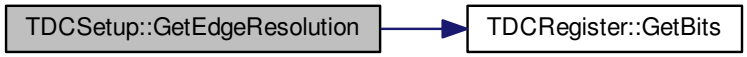
Set the adjustment of DLL taps.

Here is the call graph for this function:



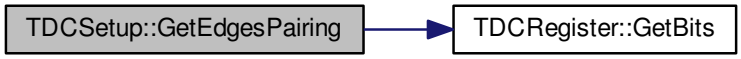
5.16.3.6 EdgeResolution TDCSetup::GetEdgeResolution ( ) const [inline]

Here is the call graph for this function:



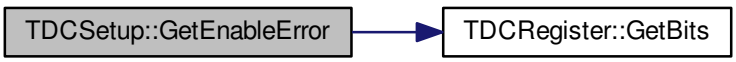
5.16.3.7 bool TDCSetup::GetEdgesPairing ( ) const [inline]

Here is the call graph for this function:



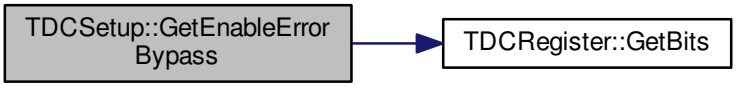
5.16.3.8 uint16_t TDCSetup::GetEnableError ( ) const [inline]

Here is the call graph for this function:



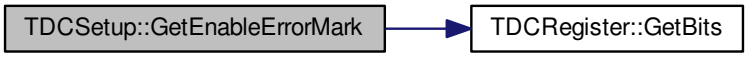
5.16.3.9 bool TDCSetup::GetEnableErrorBypass ( ) const [inline]

Here is the call graph for this function:



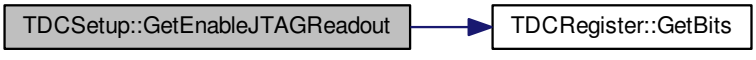
5.16.3.10 bool TDCSetup::GetEnableErrorMark ( ) const [inline]

Here is the call graph for this function:



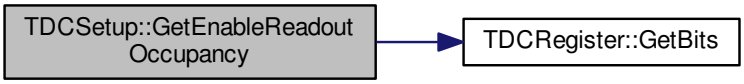
5.16.3.11 bool TDCSetup::GetEnableJTAGReadout ( ) const [inline]

Here is the call graph for this function:



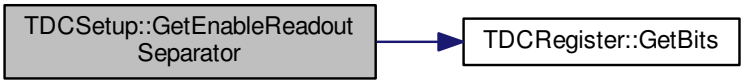
5.16.3.12 bool TDCSetup::GetEnableReadoutOccupancy ( ) const [inline]

Here is the call graph for this function:



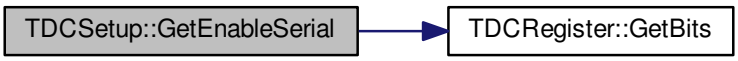
5.16.3.13 bool TDCSetup::GetEnableReadoutSeparator ( ) const [inline]

Here is the call graph for this function:



5.16.3.14 bool TDCSetup::GetEnableSerial ( ) const [inline]

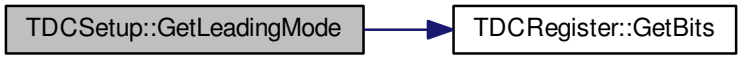
Here is the call graph for this function:



5.16.3.15 bool TDCSetup::GetLeadingMode ( ) const [inline]

Extract the status for the detection of leading edges.

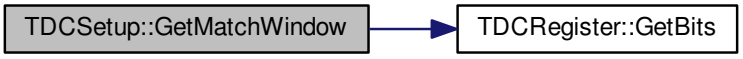
Here is the call graph for this function:



5.16.3.16 uint16_t TDCSetup::GetMatchWindow ( ) const [inline]

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

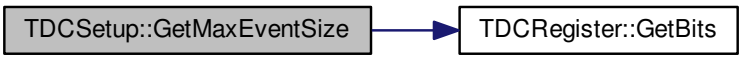
Here is the call graph for this function:



5.16.3.17 uint8_t TDCSetup::GetMaxEventSize ( ) const [inline]

Extract the maximum number of hits per event.

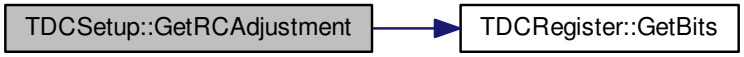
Here is the call graph for this function:



5.16.3.18 uint8_t TDCSetup::GetRCAdjustment (int *tap*) [inline]

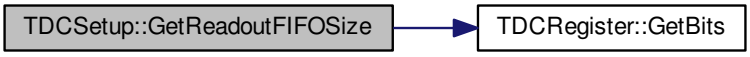
Extract the adjustment of the RC delay line.

Here is the call graph for this function:



5.16.3.19 int TDCSetup::GetReadoutFIFOSize ( ) const [inline]

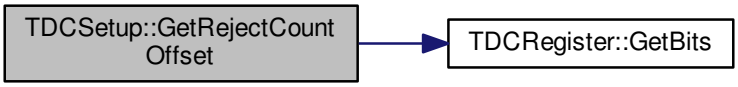
Here is the call graph for this function:



5.16.3.20 uint16_t TDCSetup::GetRejectCountOffset ( ) const [inline]

Extract the offset in reject counter.

Here is the call graph for this function:

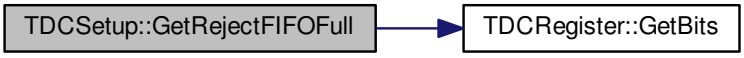


5.16.3.21 bool TDCSetup::GetRejectFIFOFull ( ) const [inline]

Are hits rejected when readout FIFO is full?

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

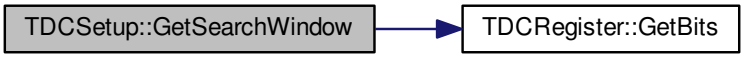
Here is the call graph for this function:



5.16.3.22 uint16_t TDCSetup::GetSearchWindow ( ) const [inline]

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

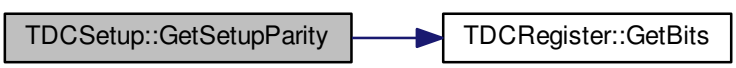
Here is the call graph for this function:



5.16.3.23 bool TDCSetup::GetSetupParity ( ) const [inline]

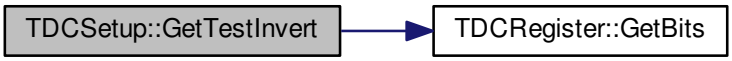
Extract the parity of setup data (should be an even parity)

Here is the call graph for this function:



5.16.3.24 bool TDCSetup::GetTestInvert ( ) const [inline]

Here is the call graph for this function:



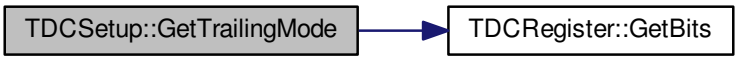
5.16.3.25 bool TDCSetup::GetTestMode ( ) const [inline]

Here is the call graph for this function:

5.16.3.26 bool TDCSetup::GetTrailingMode ( ) const [inline]

Extract the status for the detection of trailing edges.

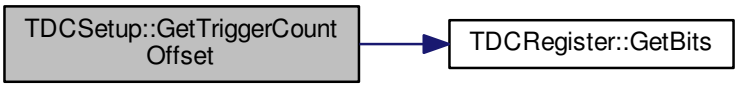
Here is the call graph for this function:



5.16.3.27 uint16_t TDCSetup::GetTriggerCountOffset ( ) const [inline]

Extract trigger time tag count offset.

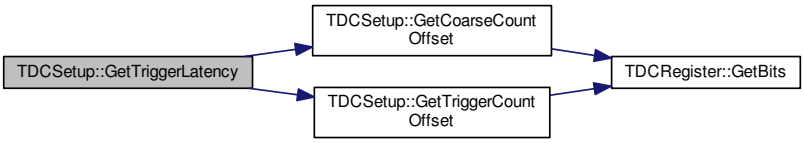
Here is the call graph for this function:



5.16.3.28 uint16_t TDCSetup::GetTriggerLatency ( ) const [inline]

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

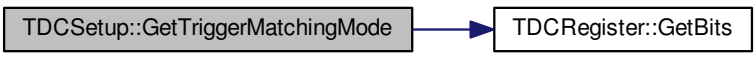
Here is the call graph for this function:



5.16.3.29 bool TDCSetup::GetTriggerMatchingMode ( ) const [inline]

Extract the enable status of trigger matching mode.

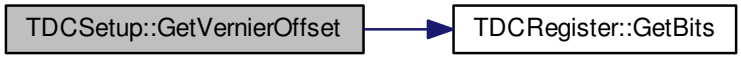
Here is the call graph for this function:



5.16.3.30 uint8_t TDCSetup::GetVernierOffset ( ) const [inline]

Extract the offset in vernier decoding.

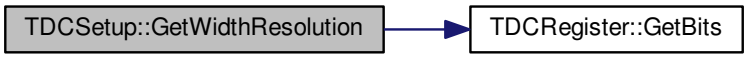
Here is the call graph for this function:



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

Extract the pulse width resolution when paired measurements are performed.

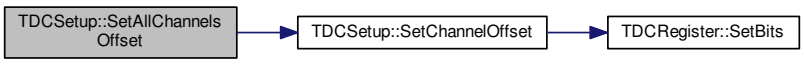
Here is the call graph for this function:



5.16.3.32 void TDCSetup::SetAllChannelsOffset (uint16_t *offset*) [inline]

Set the time offset for all channels.

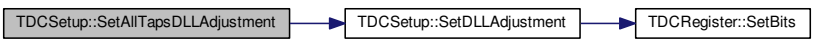
Here is the call graph for this function:



5.16.3.33 void TDCSetup::SetAllTapsDLLAdjustment (uint8_t *adj*) [inline]

Extract the adjustment of DLL taps.

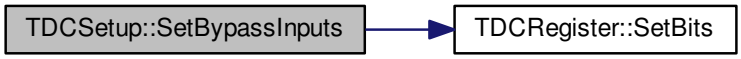
Here is the call graph for this function:



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

Select serial in and token in from bypass inputs.

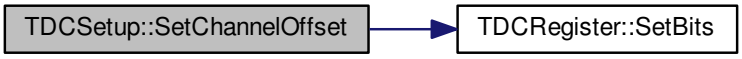
Here is the call graph for this function:



5.16.3.35 void TDCSetup::SetChannelOffset (int *channel*, uint16_t *offset*) [inline]

Set the time offset for one single channel.

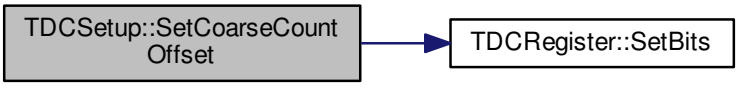
Here is the call graph for this function:



5.16.3.36 void TDCSetup::SetCoarseCountOffset (uint16_t *cco*) [inline]

Set offset for the coarse time counter.

Here is the call graph for this function:

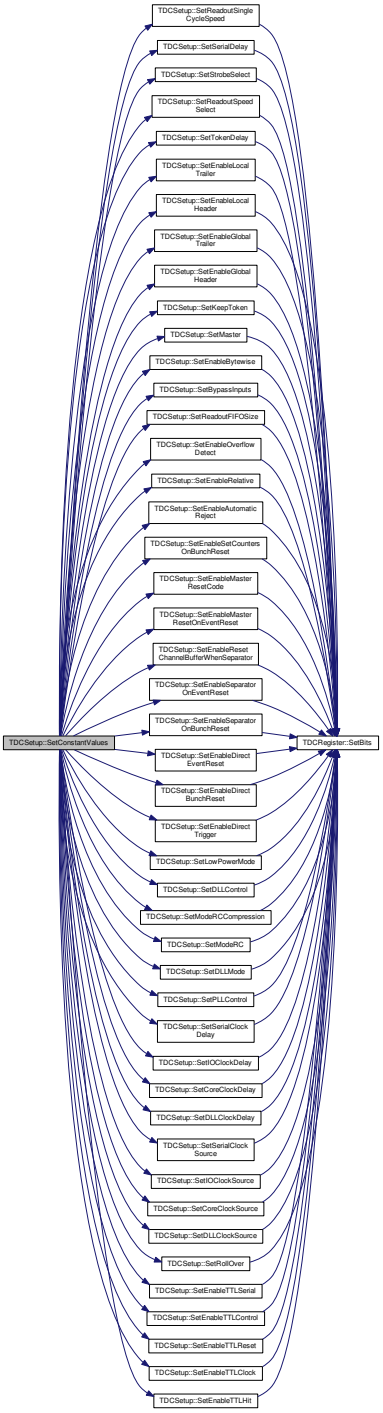


5.16.3.37 void TDCSetup::SetConstantValues ( ) [virtual]

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

Implements TDCRegister.

Here is the call graph for this function:



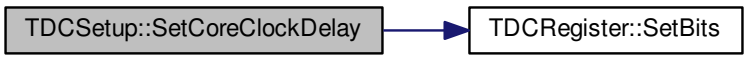
5.16.3.38 void TDCSetup::SetCoreClockDelay (const bool *delay_clock*, const uint8_t *delay*) [inline], [private]

Delay of internal core clock.

Parameters

in	*delay_clock*	Use of direct clock (0) or delayed clock (1)
in	*delay*	Delay in steps of (typically) 0.13 ns

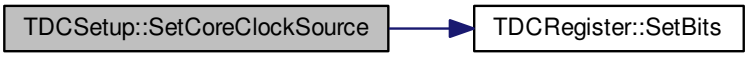
Here is the call graph for this function:



5.16.3.39 void TDCSetup::SetCoreClockSource (const CoreClockSource *ccs*) [inline],[private]

Selection of clock source for internal logic.

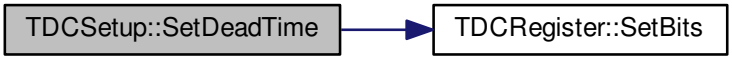
Here is the call graph for this function:



5.16.3.40 void TDCSetup::SetDeadTime (const DeadTime *dt*) [inline]

Channel dead time between hits.

Here is the call graph for this function:



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

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

Here is the call graph for this function:

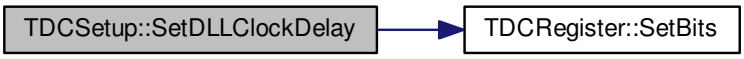
5.16.3.42 void TDCSetup::SetDLLClockDelay (const bool *delay_clock,* const uint8_t *delay*) [inline],[private]

Delay of internal DLL clock.

Parameters

in	*delay_clock*	Use of direct clock (0) or delayed clock (1)
in	*delay*	Delay in steps of (typically) 0.13 ns

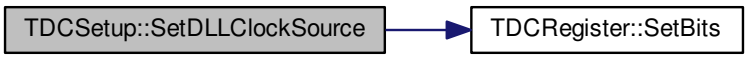
Here is the call graph for this function:



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

Selection of clock source for DLL.

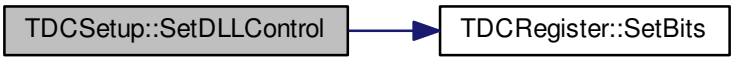
Here is the call graph for this function:



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

Control of DLL (DLL charge pump levels)

Here is the call graph for this function:



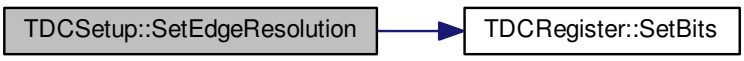
5.16.3.45 void TDCSetup::SetDLLMode (const DLLSpeedMode *dsm*) [inline],[private]

Selection of DLL speed mode.

Here is the call graph for this function:

5.16.3.46 void TDCSetup::SetEdgeResolution (const EdgeResolution *r*) [inline]

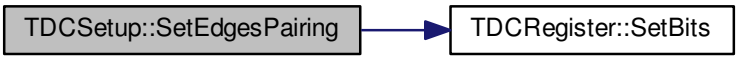
Here is the call graph for this function:



5.16.3.47 void TDCSetup::SetEdgesPairing (const bool *pair* = true) [inline]

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

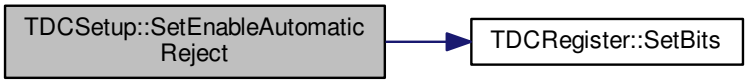
Here is the call graph for this function:



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

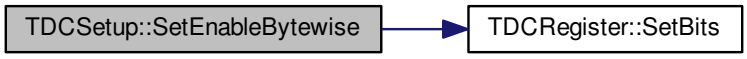
Enable of automatic rejection (should always be enabled if trigger matching mode!)

Here is the call graph for this function:



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

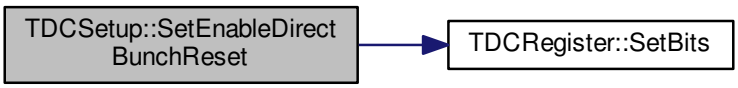
Here is the call graph for this function:



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

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

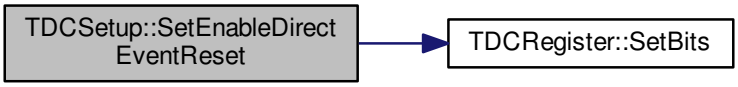
Here is the call graph for this function:



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

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

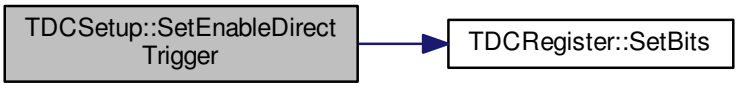
Here is the call graph for this function:



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

Enable of direct trigger input pin.

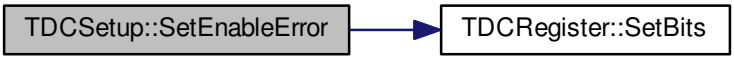
Here is the call graph for this function:



5.16.3.53 void TDCSetup::SetEnableError (const uint16_t & *err*) [inline]

Enable internal error types for generation of global error signals.

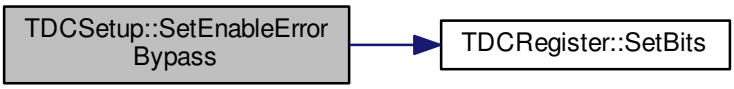
Here is the call graph for this function:



5.16.3.54 void TDCSetup::SetEnableErrorBypass (const bool *eb*) [inline]

Bypass TDC chip if global error signal is set.

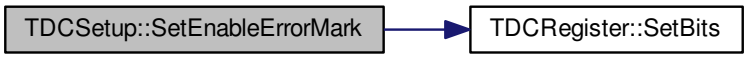
Here is the call graph for this function:



5.16.3.55 void TDCSetup::SetEnableErrorMark (const bool *em*) [inline]

Mark events with error if global error signal is set.

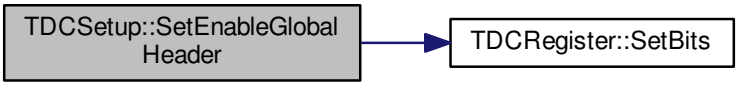
Here is the call graph for this function:



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

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

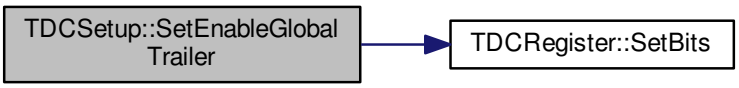
Here is the call graph for this function:



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

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

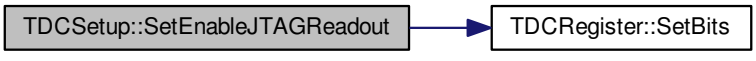
Here is the call graph for this function:



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

Enable of read-out via JTAG.

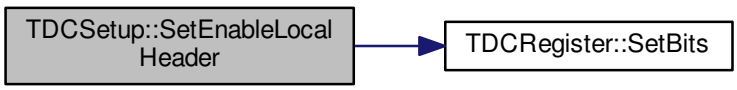
Here is the call graph for this function:



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

Enable of local headers in read-out.

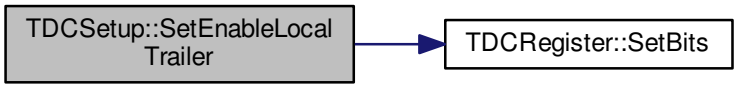
Here is the call graph for this function:



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

Enable of local trailers in read-out.

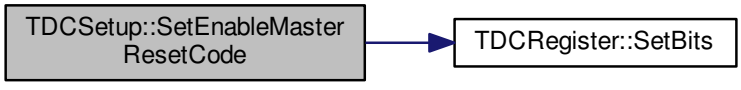
Here is the call graph for this function:



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

Enable master reset code on encoded_control.

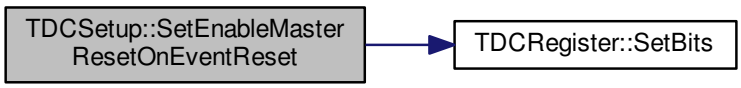
Here is the call graph for this function:



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

Enable master reset of whole TDC on event reset.

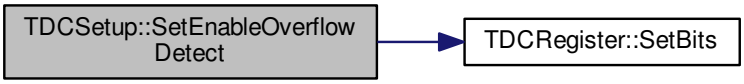
Here is the call graph for this function:



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

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

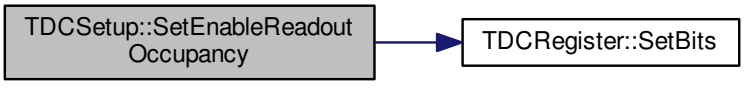
Here is the call graph for this function:



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

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

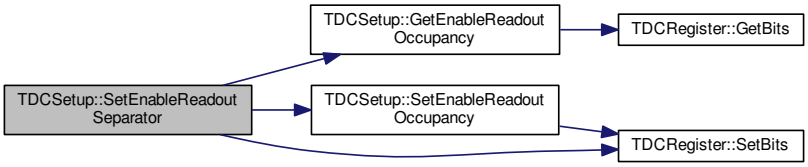
Here is the call graph for this function:



5.16.3.65 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)

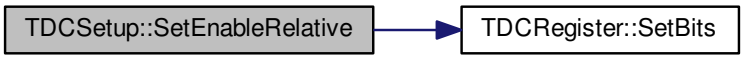
Here is the call graph for this function:



5.16.3.66 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.

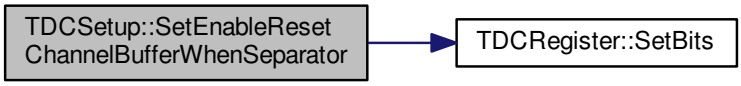
Here is the call graph for this function:



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

Enable reset channel buffers when separator.

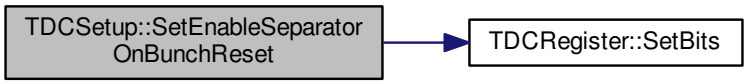
Here is the call graph for this function:



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

Enable generation of separator on bunch reset.

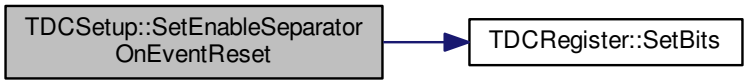
Here is the call graph for this function:



5.16.3.69 void TDCSetup::SetEnableSeparatorOnEventReset (const bool *esoer* = `true`) [inline], [private]

Enable generation of separator on event reset.

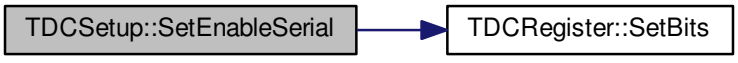
Here is the call graph for this function:



5.16.3.70 void TDCSetup::SetEnableSerial (const bool *es*) [inline]

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

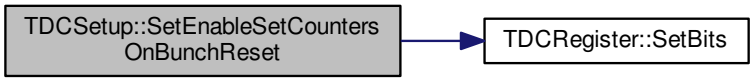
Here is the call graph for this function:



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

Enable all counters to be set on bunch count reset.

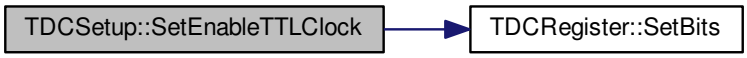
Here is the call graph for this function:



5.16.3.72 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:



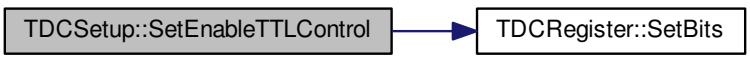
5.16.3.73 void TDCSetup::SetEnableTTLControl (const bool *tc* = `true`) [inline], [private]

Enable LV TTL inputs on control registers.

Enable LV TTL input on:

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

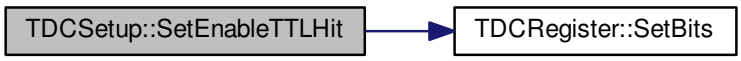
Here is the call graph for this function:



5.16.3.74 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:



5.16.3.75 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:

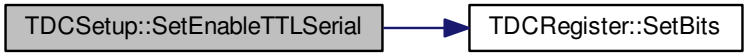
5.16.3.76 void TDCSetup::SetEnableTTLSerial (const bool *ts* = `true`) [inline], [private]

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.

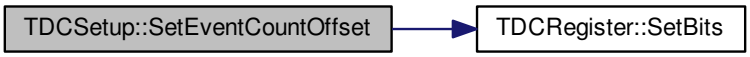
Here is the call graph for this function:



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

Set offset for the event counter.

Here is the call graph for this function:



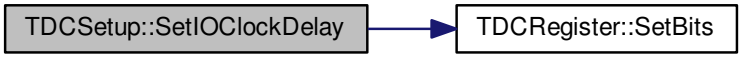
5.16.3.78 void TDCSetup::SetIOClockDelay (const bool *delay_clock,* const uint8_t *delay*) [inline], [private]

Delay of internal I/O clock.

Parameters

in	*delay_clock*	Use of direct clock (0) or delayed clock (1)
in	*delay*	Delay in steps of (typically) 0.13 ns

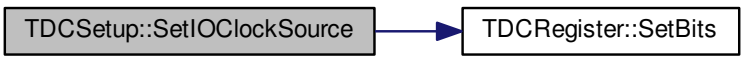
Here is the call graph for this function:



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

Selection of clock source for I/O signals.

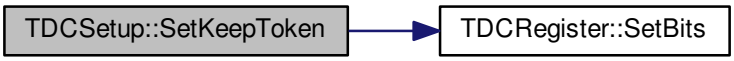
Here is the call graph for this function:



5.16.3.80 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.

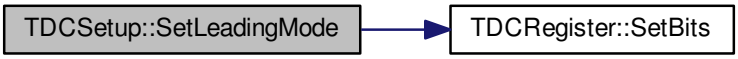
Here is the call graph for this function:



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

Enable the detection of leading edges.

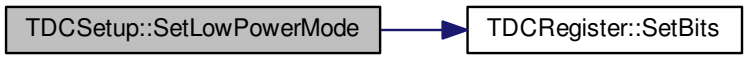
Here is the call graph for this function:



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

Low power mode of channel buffers.

Here is the call graph for this function:



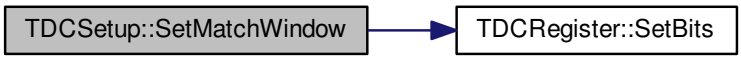
5.16.3.83 void TDCSetup::SetMaster (const bool *m* = `true`) [inline], [private]

Here is the call graph for this function:

5.16.3.84 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:

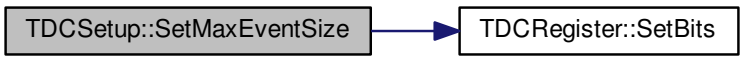


5.16.3.85 void TDCSetup::SetMaxEventSize (int *sz* = `-1`) [inline]

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 unimited.

Here is the call graph for this function:



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

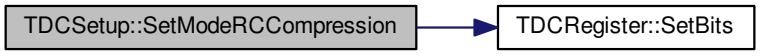
Enable of RR delay lines mode (in very high resolution mode) ; only for channels 0-4-8-12-16-20-24-28 active.

Here is the call graph for this function:

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

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

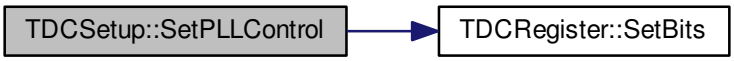
Here is the call graph for this function:



5.16.3.88 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.

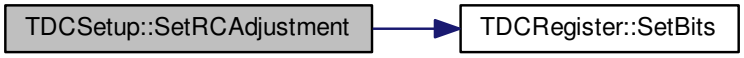
Here is the call graph for this function:



5.16.3.89 void TDCSetup::SetRCAdjustment (int *tap*, uint8_t *adj*) [inline]

Set the adjustment of the RC delay line.

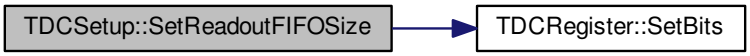
Here is the call graph for this function:



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

Effective size of readout FIFO.

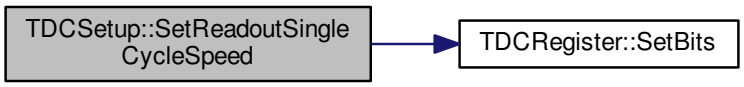
Here is the call graph for this function:



5.16.3.91 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:



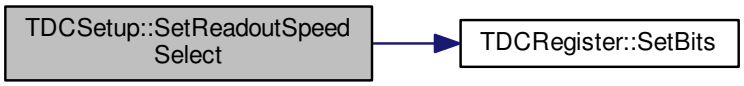
5.16.3.92 void TDCSetup::SetReadoutSpeedSelect (const ReadoutSpeed *rss* = RO_Fixed) [inline], [private]

Selection of serial read-out speed.

Parameters

in	*rss*	• 0: Selection of serial read-out speed (as defined by setup[19:17], *SetReadoutSingleCycleSpeed*) • 1: 80 Mbits/s (PLL lock required)

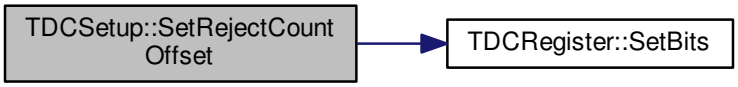
Here is the call graph for this function:



5.16.3.93 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:

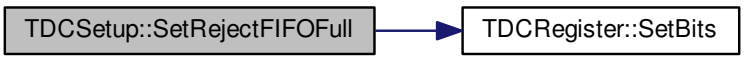


5.16.3.94 void TDCSetup::SetRejectFIFOFull (const bool *rej* = `true`) [inline]

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:



5.16.3.95 void TDCSetup::SetRollOver (const uint16_t *ro* = `0xFFF`) [inline], [private]

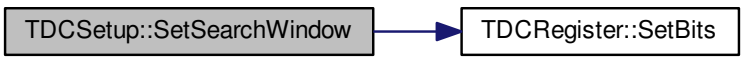
Counter roll over value, defining maximal count value from where counters will be reset to 0.

Here is the call graph for this function:

5.16.3.96 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:



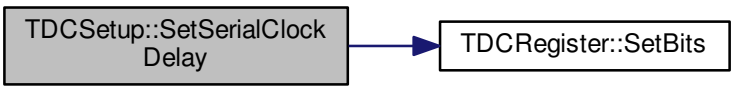
5.16.3.97 void TDCSetup::SetSerialClockDelay (const bool *delay_clock,* const uint8_t *delay*) [inline], [private]

Delay of internal serial clock.

Parameters

in	*delay_clock*	Use of direct clock (0) or delayed clock (1)
in	*delay*	Delay in steps of (typically) 0.13 ns

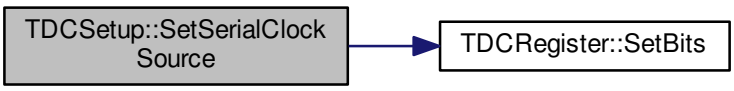
Here is the call graph for this function:



5.16.3.98 void TDCSetup::SetSerialClockSource (const SerialClockSource *scs*) [inline], [private]

Selection of source for serial clock.

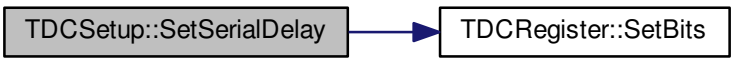
Here is the call graph for this function:



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

Programmable delay of serial input, in time unit $\sim$ 1 ns.

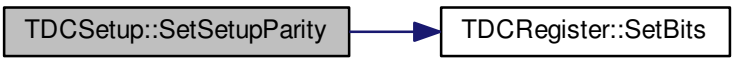
Here is the call graph for this function:



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

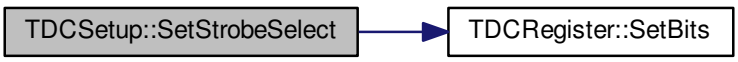
Set the parity of setup data (should be an even parity)

Here is the call graph for this function:



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

Here is the call graph for this function:



5.16.3.102 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:

5.16.3.103 void TDCSetup::SetTestMode (const bool *tm* = true) [inline]

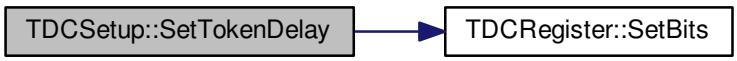
Test mode where hit data are taken from coretest. Only used during production testing.

Here is the call graph for this function:

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

Programmable delay of token input, in time unit $\sim$ 1 ns.

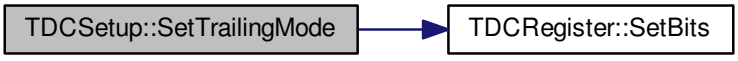
Here is the call graph for this function:



5.16.3.105 void TDCSetup::SetTrailingMode (const bool *trail* = true) [inline]

Enable/disable the detection of trailing edges.

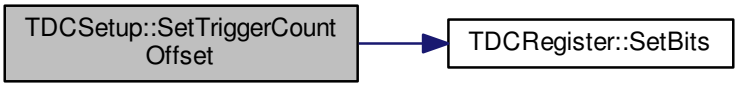
Here is the call graph for this function:



5.16.3.106 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:



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

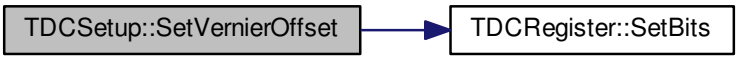
Set the enable status of trigger matching mode.

Here is the call graph for this function:

5.16.3.108 void TDCSetup::SetVernierOffset (const uint8_t *vo*) [inline]

Set the offset in vernier decoding.

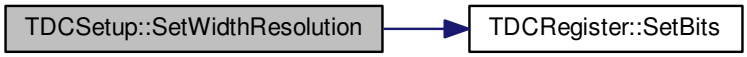
Here is the call graph for this function:



5.16.3.109 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:



5.16.4 Field Documentation

5.16.4.1 const bit TDCSetup::kCoarseCountOffset = 447 [static], [private]

5.16.4.2 const bit TDCSetup::kCoreClockDelay = 610 [static], [private]

5.16.4.3 const bit TDCSetup::kCoreClockSource = 622 [static], [private]

5.16.4.4 const bit TDCSetup::kDeadTime = 584 [static], [private]

5.16.4.5 const bit TDCSetup::kDLLClockDelay = 614 [static], [private]

5.16.4.6 const bit TDCSetup::kDLLClockSource = 624 [static], [private]

5.16.4.7 const bit TDCSetup::kDLLControl = 580 [static], [private]

5.16.4.8 const bit TDCSetup::kDLLMode = 592 [static], [private]

5.16.4.9 const bit TDCSetup::kDLLTapAdjust0 = 459 [static], [private]

5.16.4.10 const bit TDCSetup::kEnableAutomaticReject = 125 [static], [private]

5.16.4.11 const bit TDCSetup::kEnableBytewise = 37 [static], [private]

5.16.4.12 const bit TDCSetup::kEnableDirectBunchReset = 157 [static], [private]

5.16.4.13 const bit TDCSetup::kEnableDirectEventReset = 156 [static], [private]

5.16.4.14 const bit TDCSetup::kEnableDirectTrigger = 158 [static], [private]

5.16.4.15 const bit TDCSetup::kEnableError = 6 [static], [private]

5.16.4.16 const bit TDCSetup::kEnableErrorBypass = 5 [static], [private]

5.16.4.17 const bit TDCSetup::kEnableErrorMark = 4 [static], [private]

5.16.4.18 const bit TDCSetup::kEnableGlobalHeader = 34 [static], [private]

5.16.4.19 const bit TDCSetup::kEnableGlobalTrailer = 33 [static], [private]

5.16.4.20 const bit TDCSetup::kEnableJTAGReadout = 39 [static], [private]

5.16.4.21 const bit TDCSetup::kEnableLocalHeader = 32 [static], [private]

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5.16.4.50 **const bit TDCSetup::kMaxEventSize = 116** [static], [private]

5.16.4.51 **const bit TDCSetup::kModeRC = 591** [static], [private]

5.16.4.52 **const bit TDCSetup::kModeRCCompression = 590** [static], [private]

5.16.4.53 **const bit TDCSetup::kOffset0 = 438** [static], [private]

5.16.4.54 **const bit TDCSetup::kPLLControl = 594** [static], [private]

5.16.4.55 **const bit TDCSetup::kRCAdjust0 = 555** [static], [private]

5.16.4.56 **const bit TDCSetup::kReadoutFIFOSize = 45** [static], [private]

5.16.4.57 **const bit TDCSetup::kReadoutSingleCycleSpeed = 17** [static], [private]

5.16.4.58 **const bit TDCSetup::kReadoutSpeedSelect = 26** [static], [private]

5.16.4.59 **const bit TDCSetup::kRejectCountOffset = 48** [static], [private]

5.16.4.60 **const bit TDCSetup::kRejectFIFOFull = 120** [static], [private]

5.16.4.61 **const bit TDCSetup::kRollOver = 627** [static], [private]

5.16.4.62 **const bit TDCSetup::kSearchWindow = 60** [static], [private]

5.16.4.63 **const bit TDCSetup::kSelectBypassInputs = 44** [static], [private]

5.16.4.64 **const bit TDCSetup::kSerialClockDelay = 602** [static], [private]

5.16.4.65 **const bit TDCSetup::kSerialClockSource = 618** [static], [private]

5.16.4.66 **const bit TDCSetup::kSerialDelay = 20** [static], [private]

5.16.4.67 **const bit TDCSetup::kSetupParity = 646** [static], [private]

5.16.4.68 **const bit TDCSetup::kStrobeSelect = 24** [static], [private]

5.16.4.69 **const bit TDCSetup::kTDCId = 40** [static], [private]

5.16.4.70 **const bit TDCSetup::kTestInvert = 586** [static], [private]

5.16.4.71 **const bit TDCSetup::kTestMode = 587** [static], [private]

5.16.4.72 **const bit TDCSetup::kTestSelect = 0** [static], [private]

5.16.4.73 **const bit TDCSetup::kTokenDelay = 27** [static], [private]

5.16.4.74 **const bit TDCSetup::kTrailing = 588** [static], [private]

5.16.4.75 **const bit TDCSetup::kTriggerCountOffset = 138** [static], [private]

5.16.4.76 **const bit TDCSetup::kVernierOffset = 575** [static], [private]

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

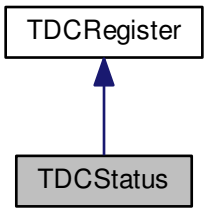
The documentation for this class was generated from the following files:

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

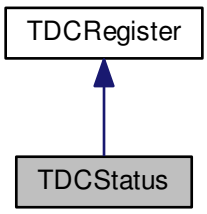
5.17 TDCStatus Class Reference

`#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

5.17.1 Detailed Description

Author

Laurent Forthomme laurent.forthomme@cern.ch

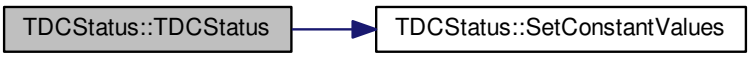
Date

27 Apr 2015

5.17.2 Constructor & Destructor Documentation

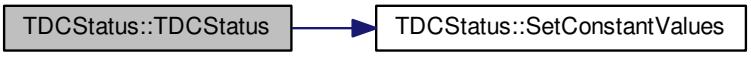
5.17.2.1 TDCStatus::TDCStatus ( ) [inline]

Here is the call graph for this function:



5.17.2.2 TDCStatus::TDCStatus (const TDCStatus & *s*) [inline]

Here is the call graph for this function:



5.17.3 Member Function Documentation

5.17.3.1 void TDCStatus::SetConstantValues ( ) [inline], [virtual]

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

Implements TDCRegister.

5.17.4 Field Documentation

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

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

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

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

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

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

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

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

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

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

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

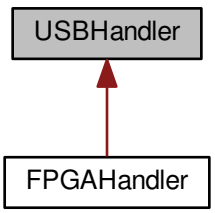
- include/TDCStatus.h

5.18 USBHandler Class Reference

Generic USB communication handler.

`#include <USBHandler.h>`

Inheritance diagram for USBHandler:



Public Member Functions

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

Protected Attributes

- bool fIsStopping

Private Attributes

- std::string fDevice
- unsigned int fVendorId

  Device's vendor identifier.
- unsigned int fProductId

  Device's product number.
- libusb_device_handle * fHandle

5.18.1 Detailed Description

Generic USB communication handler.

Date

21 Apr 2015

Author

Laurent Forthomme laurent.forthomme@cern.ch

5.18.2 Constructor & Destructor Documentation

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

5.18.2.2 USBHandler::USBHandler (const unsigned int *vendor_id*, const unsigned int *product_id*)

5.18.2.3 USBHandler::~USBHandler ( ) [virtual]

5.18.3 Member Function Documentation

5.18.3.1 void USBHandler::DumpDevice (libusb_device * *dev*, int *verb* = 1, std::ostream & *out* = std::cout)

5.18.3.2 uint32_t USBHandler::Fetch (uint8_t *size*) const

Receive a word from the USB device.

5.18.3.3 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:

5.18.3.4 void USBHandler::Reset ( ) const

5.18.3.5 void USBHandler::Write (uint32_t *word,* uint8_t *size*) const

Write a word to the USB device.

5.18.4 Field Documentation

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

5.18.4.2 libusb_device_handle* USBHandler::fHandle [private]

5.18.4.3 bool USBHandler::fIsStopping [protected]

5.18.4.4 unsigned int USBHandler::fProductId [private]

Device's product number.

5.18.4.5 unsigned int USBHandler::fVendorId [private]

Device's vendor identifier.

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, 27
~FileReader
FileReader, 25
~Message
Message, 33
~Messenger
Messenger, 36
~Socket
Socket, 43
~SocketMessage
SocketMessage, 52
~TDC
TDC, 55
~TDCEvent
TDCEvent, 69
~TDCRegister
TDCRegister, 73
~USBHandler
USBHandler, 128

AcceptConnections
Socket, 43
AddClient
Messenger, 36
Announce
Client, 17

Bind
Socket, 44
bit
TDCRegister, 73
Broadcast
Messenger, 36

CLIENT
Socket communication objects, 8
ChannelSelectError
HPTDC chip control, 11
CheckFirmwareVersion
TDC, 56
Clear
TDCRegister, 74
Client, 15
~Client, 17
Announce, 17
Client, 17
Connect, 18
Disconnect, 18
fClientId, 20
fIsConnected, 20
GetType, 19
ParseMessage, 19
Receive, 19
Send, 19
SendAndReceive, 20
CloseFile
FPGAHandler, 28
CoarseError
HPTDC chip control, 11
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, 10
DLL_320MHz
HPTDC chip control, 10
DLL_40MHz
HPTDC chip control, 10
DLL_Illegal
HPTDC chip control, 10
DLL_aux_clock
HPTDC chip control, 10
DLL_clock_40
HPTDC chip control, 10
DLL_pll_clock_160
HPTDC chip control, 10

DLL_pll_clock_320
HPTDC chip control, 10
DLL_pll_clock_40
HPTDC chip control, 10
DLLClockSource
HPTDC chip control, 10
DLLSpeedMode
HPTDC chip control, 10
DT_100ns
HPTDC chip control, 10
DT_10ns
HPTDC chip control, 10
DT_30ns
HPTDC chip control, 10
DT_5ns
HPTDC chip control, 10
DeadTime
HPTDC chip control, 10
Debug
HPTDC chip control, 11
Decode
HTTPMessage, 31
Description
Exception, 22
DisableAllChannels
TDCControl, 63
DisableChannel
TDCControl, 63
Disconnect
Client, 18
Messenger, 37
DisconnectClient
Messenger, 38
Dump
Exception, 22
HTTPMessage, 31
Message, 33
SocketMessage, 52
TDCControl, 64
TDCSetup, 84
DumpConnected
Socket, 44
DumpDevice
USBHandler, 128
DumpRegister
TDCRegister, 74

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, 10
EnableAllChannels
TDCControl, 64
EnableChannel
TDCControl, 64
EnablePattern
HPTDC chip control, 11
EnabledError
HPTDC chip control, 11
Encode
HTTPMessage, 31
Error
HPTDC chip control, 11
ErrorNumber
Exception, 22
ErrorState
FPGAHandler, 28
EventType
HPTDC chip control, 11
Exception, 20
~Exception, 21
Description, 22
Dump, 22
ErrorNumber, 22
Exception, 21
fDescription, 22
fErrorNumber, 22
fFrom, 22
fType, 22
From, 22
Type, 22
TypeString, 22

fAddress
Socket, 47
fBS
TDC, 58
fBuffer
Socket, 47
fClientId
Client, 20
fControl
TDC, 58
fDescription
Exception, 22
fDevice
USBHandler, 129
fErrorNumber
Exception, 22
fFile
FileReader, 25
fFilename
FPGAHandler, 29
fFrom

Exception, 22
fHandle
USBHandler, 129
fHeader
FileReader, 25
fId
TDC, 58
fIsConnected
Client, 20
fIsFileOpen
FPGAHandler, 29
fIsStopping
USBHandler, 129
fIsTDCInReadout
FPGAHandler, 29
fMaster
Socket, 47
fMessage
SocketMessage, 54
fNumAttempts
Messenger, 41
fNumWords
TDCRegister, 75
fOriginalString
HTTPMessage, 32
fOutput
FPGAHandler, 29
FPGA board control, 7
FPGAHandler, 25
∼FPGAHandler, 27
CloseFile, 28
ErrorState, 28
fFilename, 29
fIsFileOpen, 29
fIsTDCInReadout, 29
fOutput, 29
FPGAHandler, 27
fTDC, 29
GetFilename, 28
GetTDC, 28
GetType, 28
OpenFile, 28
ReadBuffer, 28
SetTDCSetup, 29
Stop, 29
fPort
Socket, 47
fProductId
USBHandler, 129
fReadFds
Socket, 47
fSetup
TDC, 58
fSetupCollection
FileReader, 25
fSocketId
Socket, 47
fSocketsConnected
Socket, 47
fStatus
TDC, 58
fString
Message, 33
fTDC
FPGAHandler, 29
fType
Exception, 22
fUSB
TDC, 58
fVendorId
USBHandler, 129
fWS
HTTPMessage, 32
Messenger, 41
fWord
TDCEvent, 71
TDCRegister, 75
fWordSize
TDCRegister, 75
Fetch
USBHandler, 128
FetchEvents
TDC, 56
FetchMessage
Socket, 44
file_header_t, 23
magic, 23
num_hptdc, 23
run_id, 23
spill_id, 23
FileReader, 23
∼FileReader, 25
fFile, 25
fHeader, 25
fSetupCollection, 25
FileReader, 24
GetNextEvent, 25
GetNumTDCs, 25
From
Exception, 22

GetBits
TDCRegister, 74
GetBunchId
TDCEvent, 69
GetChannelOffset
TDCSetup, 85
GetCoarseCountOffset
TDCSetup, 85
GetDLLAdjustment
TDCSetup, 86
GetDLLReset
TDCControl, 64
GetDeadTime
TDCSetup, 86
GetEdgeResolution
TDCSetup, 86

GetEdgesPairing
TDCSetup, 87
GetEnableError
TDCSetup, 87
GetEnableErrorBypass
TDCSetup, 87
GetEnableErrorMark
TDCSetup, 88
GetEnableJTAGReadout
TDCSetup, 88
GetEnablePattern
TDCControl, 65
GetEnableReadoutOccupancy
TDCSetup, 88
GetEnableReadoutSeparator
TDCSetup, 89
GetEnableSerial
TDCSetup, 89
GetErrorFlags
TDCEvent, 69
GetEventId
TDCEvent, 69
GetFilename
FPGAHandler, 28
GetGlobalReset
TDCControl, 65
GetIntValue
SocketMessage, 52
GetKey
HTTPMessage, 31
Message, 33
SocketMessage, 52
GetLeadingMode
TDCSetup, 89
GetLeadingTime
TDCEvent, 70
GetMatchWindow
TDCSetup, 90
GetMaxEventSize
TDCSetup, 90
GetNextEvent
FileReader, 25
GetNumTDCs
FileReader, 25
GetNumWords
TDCRegister, 74
GetPLLReset
TDCControl, 65
GetPort
Socket, 44
GetRCAdjustment
TDCSetup, 90
GetReadoutFIFOSize
TDCSetup, 91
GetRejectCountOffset
TDCSetup, 91
GetRejectFIFOFull
TDCSetup, 91
GetSearchWindow
TDCSetup, 92
GetSetupParity
TDCSetup, 92
GetSetupRegister
TDC, 56
GetSocketId
Socket, 45
GetSocketType
Socket, 45
GetString
Message, 33
SocketMessage, 52
GetTDC
FPGAHandler, 28
GetTDCId
TDCEvent, 70
GetTestInvert
TDCSetup, 92
GetTestMode
TDCSetup, 93
GetTrailingMode
TDCSetup, 93
GetTrailingTime
TDCEvent, 70
GetTriggerCountOffset
TDCSetup, 93
GetTriggerLatency
TDCSetup, 94
GetTriggerMatchingMode
TDCSetup, 94
GetType
Client, 19
FPGAHandler, 28
Messenger, 38
TDCEvent, 70
GetValue
SocketMessage, 52
GetVectorValue
SocketMessage, 52
GetVernierOffset
TDCSetup, 94
GetWidth
TDCEvent, 71
GetWidthResolution
TDCSetup, 95
GetWord
TDCRegister, 74
GetWordCount
TDCEvent, 71
GroupHeader
HPTDC chip control, 11
GroupTrailer
HPTDC chip control, 11

HPTDC chip control, 9
ChannelSelectError, 11
CoarseError, 11
ControlParityError, 11

Core_aux_clock, 10
Core_clock_40, 10
Core_pll_clock_160, 10
Core_pll_clock_80, 10
CoreClockSource, 10
DLL_160MHz, 10
DLL_320MHz, 10
DLL_40MHz, 10
DLL_Illegal, 10
DLL_aux_clock, 10
DLL_clock_40, 10
DLL_pll_clock_160, 10
DLL_pll_clock_320, 10
DLL_pll_clock_40, 10
DLLClockSource, 10
DLLSpeedMode, 10
DT_100ns, 10
DT_10ns, 10
DT_30ns, 10
DT_5ns, 10
DeadTime, 10
Debug, 11
E_100ps, 11
E_12p5ns, 11
E_1p6ns, 11
E_200ps, 11
E_3p12ns, 11
E_400ps, 11
E_6p25ns, 11
E_800ps, 11
EdgeResolution, 10
EnablePattern, 11
EnabledError, 11
Error, 11
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, 11
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, 11
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, 32
fWS, 32
GetKey, 31
HTTPMessage, 31

INVALID
Socket communication objects, 8
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, 11
Init
USBHandler, 128
Invalid
HPTDC chip control, 11
IsFromWeb
Message, 33
IsWebSocket
Socket, 45

JTAGInstructionParityError
HPTDC chip control, 11

kAuxClock
TDCBoundaryScan, 60
kBunchReset
TDCBoundaryScan, 60
kClk
TDCBoundaryScan, 60
kCoarseCountOffset
TDCSetup, 122
kControlParity
TDCControl, 67
kCoreClockDelay
TDCSetup, 122
kCoreClockSource
TDCSetup, 122
kDLLClockDelay
TDCSetup, 122
kDLLClockSource
TDCSetup, 122
kDLLControl
TDCSetup, 122
kDLLLock
TDCStatus, 127
kDLLMode
TDCSetup, 122
kDLLReset
TDCControl, 67
kDLLTapAdjust0
TDCSetup, 122
kDataReady
TDCBoundaryScan, 60
kDeadTime
TDCSetup, 122
kEnableAutomaticReject
TDCSetup, 122
kEnableBytewise
TDCSetup, 122
kEnableChannel
TDCControl, 67
kEnableDirectBunchReset
TDCSetup, 122
kEnableDirectEventReset
TDCSetup, 122
kEnableDirectTrigger
TDCSetup, 122
kEnableError
TDCSetup, 122
kEnableErrorBypass
TDCSetup, 122
kEnableErrorMark
TDCSetup, 122
kEnableGlobalHeader
TDCSetup, 122
kEnableGlobalTrailer
TDCSetup, 122
kEnableJTAGReadout
TDCSetup, 122
kEnableLocalHeader
TDCSetup, 122
kEnableLocalTrailer
TDCSetup, 122
kEnableMasterResetCode
TDCSetup, 123
kEnableMasterResetOnEventReset
TDCSetup, 123
kEnableMatching
TDCSetup, 123
kEnableOverflowDetect
TDCSetup, 123
kEnablePair
TDCSetup, 123
kEnablePattern
TDCControl, 67
kEnableReadoutOccupancy
TDCSetup, 123
kEnableReadoutSeparator
TDCSetup, 123
kEnableRelative
TDCSetup, 123
kEnableResetChannelBufferWhenSeparator
TDCSetup, 123
kEnableSeparatorOnBunchReset
TDCSetup, 123
kEnableSeparatorOnEventReset
TDCSetup, 123
kEnableSerial
TDCSetup, 123
kEnableSetCountersOnBunchReset
TDCSetup, 123
kEnableTTLClock
TDCSetup, 123
kEnableTTLControl
TDCSetup, 123
kEnableTTLHit
TDCSetup, 123
kEnableTTLReset
TDCSetup, 123
kEnableTTLSerial
TDCSetup, 123
kEncodedControl
TDCBoundaryScan, 60
kError
TDCBoundaryScan, 60
TDCStatus, 127
kEventCountOffset

TDCSetup, 123
kEventReset
TDCBoundaryScan, 60
kGetData
TDCBoundaryScan, 61
kGlobalReset
TDCControl, 67
kHaveToken
TDCStatus, 127
kHit
TDCBoundaryScan, 61
kIOClockDelay
TDCSetup, 123
kIOClockSource
TDCSetup, 123
kKeepToken
TDCSetup, 123
kL1Occupancy
TDCStatus, 127
kLeading
TDCSetup, 123
kLeadingResolution
TDCSetup, 123
kLowPowerMode
TDCSetup, 123
kMaster
TDCSetup, 123
kMatchWindow
TDCSetup, 123
kMaxEventSize
TDCSetup, 123
kModeRC
TDCSetup, 124
kModeRCCompression
TDCSetup, 124
kOffset0
TDCSetup, 124
kPLLControl
TDCSetup, 124
kPLLReset
TDCControl, 67
kParallelDataOut
TDCBoundaryScan, 61
kParallelEnable
TDCBoundaryScan, 61
kRCAdjust0
TDCSetup, 124
kReadoutFIFOEmpty
TDCStatus, 127
kReadoutFIFOFull
TDCStatus, 127
kReadoutFIFOOccupancy
TDCStatus, 127
kReadoutFIFOSize
TDCSetup, 124
kReadoutSingleCycleSpeed
TDCSetup, 124
kReadoutSpeedSelect
TDCSetup, 124
kRejectCountOffset
TDCSetup, 124
kRejectFIFOFull
TDCSetup, 124
kReset
TDCBoundaryScan, 61
kRollOver
TDCSetup, 124
kSearchWindow
TDCSetup, 124
kSelectBypassInputs
TDCSetup, 124
kSerialBypassIn
TDCBoundaryScan, 61
kSerialClockDelay
TDCSetup, 124
kSerialClockSource
TDCSetup, 124
kSerialDelay
TDCSetup, 124
kSerialIn
TDCBoundaryScan, 61
kSerialOut
TDCBoundaryScan, 61
kSetupParity
TDCSetup, 124
kStrobeOut
TDCBoundaryScan, 61
kStrobeSelect
TDCSetup, 124
kTDCId
TDCSetup, 124
kTest
TDCBoundaryScan, 61
kTestInvert
TDCSetup, 124
kTestMode
TDCSetup, 124
kTestSelect
TDCSetup, 124
kTokenBypassIn
TDCBoundaryScan, 61
kTokenDelay
TDCSetup, 124
kTokenIn
TDCBoundaryScan, 61
kTokenOut
TDCBoundaryScan, 61
kTrailing
TDCSetup, 124
kTrigger
TDCBoundaryScan, 61
kTriggerCountOffset
TDCSetup, 124
kTriggerFIFOEmpty
TDCStatus, 127
kTriggerFIFOFull

TDCStatus, 127
kTriggerFIFOOccupancy
TDCStatus, 127
kVernierOffset
TDCSetup, 124
kWidthSelect
TDCSetup, 124

L1BufferParityError
HPTDC chip control, 11
LeadingEdge
HPTDC chip control, 11
Listen
Socket, 45

MASTER
Socket communication objects, 8
magic
file_header_t, 23
Message, 32
∼Message, 33
Dump, 33
fString, 33
GetKey, 33
GetString, 33
IsFromWeb, 33
Message, 33
Messenger, 34
∼Messenger, 36
AddClient, 36
Broadcast, 36
Connect, 37
Disconnect, 37
DisconnectClient, 38
fNumAttempts, 41
fWS, 41
GetType, 38
Messenger, 35
ProcessMessage, 38
Receive, 39
Send, 39
StartAcquisition, 40
StopAcquisition, 40
SwitchClientType, 40

num_hptdc
file_header_t, 23

Object
SocketMessage, 53
OpenFile
FPGAHandler, 28
operator=
TDCRegister, 74

ParseMessage
Client, 19
PrepareConnection
Socket, 45
ProcessMessage
Messenger, 38

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
ReadBuffer
FPGAHandler, 28
ReadConfiguration
TDC, 56
ReadRegister
TDC, 56
ReadStatus
TDC, 57
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
Reset
USBHandler, 129
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, 39
SendAndReceive
Client, 20
SendConfiguration
TDC, 57
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, 95
SetAllTapsDLLAdjustment
TDCSetup, 95
SetBits
TDCRegister, 74
SetBypassInputs
TDCSetup, 95
SetChannelOffset
TDCSetup, 96
SetCoarseCountOffset
TDCSetup, 96
SetConstantValues
TDCBoundaryScan, 60
TDCControl, 66
TDCRegister, 74
TDCSetup, 96
TDCStatus, 126
SetControlParity
TDCControl, 66
SetCoreClockDelay
TDCSetup, 97
SetCoreClockSource
TDCSetup, 98
SetDLLAdjustment
TDCSetup, 98
SetDLLClockDelay
TDCSetup, 99
SetDLLClockSource
TDCSetup, 99
SetDLLControl
TDCSetup, 99
SetDLLMode
TDCSetup, 100
SetDLLReset
TDCControl, 66
SetDeadTime
TDCSetup, 98
SetEdgeResolution
TDCSetup, 100
SetEdgesPairing
TDCSetup, 100
SetEnableAutomaticReject
TDCSetup, 101
SetEnableBytewise
TDCSetup, 101
SetEnableDirectBunchReset
TDCSetup, 101
SetEnableDirectEventReset
TDCSetup, 102
SetEnableDirectTrigger
TDCSetup, 102
SetEnableError
TDCSetup, 102
SetEnableErrorBypass
TDCSetup, 103
SetEnableErrorMark
TDCSetup, 103
SetEnableGlobalHeader
TDCSetup, 103
SetEnableGlobalTrailer
TDCSetup, 104
SetEnableJTAGReadout
TDCSetup, 104
SetEnableLocalHeader
TDCSetup, 104
SetEnableLocalTrailer
TDCSetup, 105
SetEnableMasterResetCode
TDCSetup, 105
SetEnableMasterResetOnEventReset
TDCSetup, 105
SetEnableOverflowDetect
TDCSetup, 106
SetEnablePattern
TDCControl, 67
SetEnableReadoutOccupancy
TDCSetup, 106
SetEnableReadoutSeparator
TDCSetup, 106
SetEnableRelative
TDCSetup, 107
SetEnableResetChannelBufferWhenSeparator
TDCSetup, 107
SetEnableSeparatorOnBunchReset
TDCSetup, 107
SetEnableSeparatorOnEventReset

TDCSetup, 108
SetEnableSerial
TDCSetup, 108
SetEnableSetCountersOnBunchReset
TDCSetup, 108
SetEnableTTLClock
TDCSetup, 109
SetEnableTTLControl
TDCSetup, 109
SetEnableTTLHit
TDCSetup, 109
SetEnableTTLReset
TDCSetup, 110
SetEnableTTLSerial
TDCSetup, 110
SetEventCountOffset
TDCSetup, 111
SetGlobalReset
TDCControl, 67
SetIOClockDelay
TDCSetup, 111
SetIOClockSource
TDCSetup, 111
SetKeepToken
TDCSetup, 112
SetKeyValue
SocketMessage, 53, 54
SetLeadingMode
TDCSetup, 112
SetLowPowerMode
TDCSetup, 112
SetMaster
TDCSetup, 113
SetMatchWindow
TDCSetup, 113
SetMaxEventSize
TDCSetup, 113
SetModeRC
TDCSetup, 114
SetModeRCCompression
TDCSetup, 114
SetPLLControl
TDCSetup, 114
SetPLLReset
TDCControl, 67
SetPort
Socket, 46
SetRCAdjustment
TDCSetup, 115
SetReadoutFIFOSize
TDCSetup, 115
SetReadoutSingleCycleSpeed
TDCSetup, 115
SetReadoutSpeedSelect
TDCSetup, 116
SetRejectCountOffset
TDCSetup, 116
SetRejectFIFOFull
TDCSetup, 116
SetRollOver
TDCSetup, 117
SetSearchWindow
TDCSetup, 117
SetSerialClockDelay
TDCSetup, 117
SetSerialClockSource
TDCSetup, 118
SetSerialDelay
TDCSetup, 118
SetSetupParity
TDCSetup, 118
SetSetupRegister
TDC, 57
SetSocketId
Socket, 46
SetStrobeSelect
TDCSetup, 119
SetTDCSetup
FPGAHandler, 29
SetTestInvert
TDCSetup, 119
SetTestMode
TDCSetup, 119
SetTokenDelay
TDCSetup, 120
SetTrailingMode
TDCSetup, 120
SetTriggerCountOffset
TDCSetup, 120
SetTriggerMatchingMode
TDCSetup, 121
SetVernierOffset
TDCSetup, 121
SetWidthResolution
TDCSetup, 121
SetWord
TDCEvent, 71
TDCRegister, 74
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, 8
CLIENT, 8
DETECTOR, 8
INVALID, 8
MASTER, 8
SocketType, 8
WEBSOCKET_CLIENT, 8
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, 8
SoftReset
TDC, 57
spill_id
file_header_t, 23
Start
Socket, 46
StartAcquisition
Messenger, 40
Stop
FPGAHandler, 29
Socket, 46
StopAcquisition
Messenger, 40
String
SocketMessage, 54
SwitchClientType
Messenger, 40

TDC, 54
∼TDC, 55
CheckFirmwareVersion, 56
fBS, 58
fControl, 58
fId, 58
fSetup, 58
fStatus, 58
fUSB, 58
FetchEvents, 56
GetSetupRegister, 56
ReadConfiguration, 56
ReadRegister, 56
ReadStatus, 57
SendConfiguration, 57
SetSetupRegister, 57
SoftReset, 57
TDC, 55
WriteRegister, 57
TDCBoundaryScan, 58
kAuxClock, 60
kBunchReset, 60
kClk, 60
kDataReady, 60
kEncodedControl, 60
kError, 60
kEventReset, 60
kGetData, 61
kHit, 61
kParallelDataOut, 61
kParallelEnable, 61
kReset, 61
kSerialBypassIn, 61
kSerialIn, 61
kSerialOut, 61
kStrobeOut, 61
kTest, 61
kTokenBypassIn, 61
kTokenIn, 61
kTokenOut, 61
kTrigger, 61
SetConstantValues, 60
TDCBoundaryScan, 60
TDCControl, 61
DisableAllChannels, 63
DisableChannel, 63
Dump, 64
EnableAllChannels, 64
EnableChannel, 64
GetDLLReset, 64
GetEnablePattern, 65
GetGlobalReset, 65
GetPLLReset, 65
kControlParity, 67
kDLLReset, 67
kEnableChannel, 67
kEnablePattern, 67
kGlobalReset, 67
kPLLReset, 67
SetConstantValues, 66
SetControlParity, 66
SetDLLReset, 66

SetEnablePattern, 67
SetGlobalReset, 67
SetPLLReset, 67
TDCControl, 63
TDCEvent, 68
~TDCEvent, 69
fWord, 71
GetBunchId, 69
GetErrorFlags, 69
GetEventId, 69
GetLeadingTime, 70
GetTDCId, 70
GetTrailingTime, 70
GetType, 70
GetWidth, 71
GetWordCount, 71
SetWord, 71
TDCEvent, 69
TDCHeader
HPTDC chip control, 11
TDCRegister, 71
~TDCRegister, 73
bit, 73
Clear, 74
DumpRegister, 74
fNumWords, 75
fWord, 75
fWordSize, 75
GetBits, 74
GetNumWords, 74
GetWord, 74
operator=, 74
SetBits, 74
SetConstantValues, 74
SetWord, 74
TDCRegister, 73
word_t, 73
TDCSetup, 75
Dump, 84
GetChannelOffset, 85
GetCoarseCountOffset, 85
GetDLLAdjustment, 86
GetDeadTime, 86
GetEdgeResolution, 86
GetEdgesPairing, 87
GetEnableError, 87
GetEnableErrorBypass, 87
GetEnableErrorMark, 88
GetEnableJTAGReadout, 88
GetEnableReadoutOccupancy, 88
GetEnableReadoutSeparator, 89
GetEnableSerial, 89
GetLeadingMode, 89
GetMatchWindow, 90
GetMaxEventSize, 90
GetRCAdjustment, 90
GetReadoutFIFOSize, 91
GetRejectCountOffset, 91
GetRejectFIFOFull, 91
GetSearchWindow, 92
GetSetupParity, 92
GetTestInvert, 92
GetTestMode, 93
GetTrailingMode, 93
GetTriggerCountOffset, 93
GetTriggerLatency, 94
GetTriggerMatchingMode, 94
GetVernierOffset, 94
GetWidthResolution, 95
kCoarseCountOffset, 122
kCoreClockDelay, 122
kCoreClockSource, 122
kDLLClockDelay, 122
kDLLClockSource, 122
kDLLControl, 122
kDLLMode, 122
kDLLTapAdjust0, 122
kDeadTime, 122
kEnableAutomaticReject, 122
kEnableBytewise, 122
kEnableDirectBunchReset, 122
kEnableDirectEventReset, 122
kEnableDirectTrigger, 122
kEnableError, 122
kEnableErrorBypass, 122
kEnableErrorMark, 122
kEnableGlobalHeader, 122
kEnableGlobalTrailer, 122
kEnableJTAGReadout, 122
kEnableLocalHeader, 122
kEnableLocalTrailer, 122
kEnableMasterResetCode, 123
kEnableMasterResetOnEventReset, 123
kEnableMatching, 123
kEnableOverflowDetect, 123
kEnablePair, 123
kEnableReadoutOccupancy, 123
kEnableReadoutSeparator, 123
kEnableRelative, 123
kEnableResetChannelBufferWhenSeparator, 123
kEnableSeparatorOnBunchReset, 123
kEnableSeparatorOnEventReset, 123
kEnableSerial, 123
kEnableSetCountersOnBunchReset, 123
kEnableTTLClock, 123
kEnableTTLControl, 123
kEnableTTLHit, 123
kEnableTTLReset, 123
kEnableTTLSerial, 123
kEventCountOffset, 123
kIOClockDelay, 123
kIOClockSource, 123
kKeepToken, 123
kLeading, 123
kLeadingResolution, 123
kLowPowerMode, 123

kMaster, 123
kMatchWindow, 123
kMaxEventSize, 123
kModeRC, 124
kModeRCCompression, 124
kOffset0, 124
kPLLControl, 124
kRCAdjust0, 124
kReadoutFIFOSize, 124
kReadoutSingleCycleSpeed, 124
kReadoutSpeedSelect, 124
kRejectCountOffset, 124
kRejectFIFOFull, 124
kRollOver, 124
kSearchWindow, 124
kSelectBypassInputs, 124
kSerialClockDelay, 124
kSerialClockSource, 124
kSerialDelay, 124
kSetupParity, 124
kStrobeSelect, 124
kTDCId, 124
kTestInvert, 124
kTestMode, 124
kTestSelect, 124
kTokenDelay, 124
kTrailing, 124
kTriggerCountOffset, 124
kVernierOffset, 124
kWidthSelect, 124
SetAllChannelsOffset, 95
SetAllTapsDLLAdjustment, 95
SetBypassInputs, 95
SetChannelOffset, 96
SetCoarseCountOffset, 96
SetConstantValues, 96
SetCoreClockDelay, 97
SetCoreClockSource, 98
SetDLLAdjustment, 98
SetDLLClockDelay, 99
SetDLLClockSource, 99
SetDLLControl, 99
SetDLLMode, 100
SetDeadTime, 98
SetEdgeResolution, 100
SetEdgesPairing, 100
SetEnableAutomaticReject, 101
SetEnableBytewise, 101
SetEnableDirectBunchReset, 101
SetEnableDirectEventReset, 102
SetEnableDirectTrigger, 102
SetEnableError, 102
SetEnableErrorBypass, 103
SetEnableErrorMark, 103
SetEnableGlobalHeader, 103
SetEnableGlobalTrailer, 104
SetEnableJTAGReadout, 104
SetEnableLocalHeader, 104
SetEnableLocalTrailer, 105
SetEnableMasterResetCode, 105
SetEnableMasterResetOnEventReset, 105
SetEnableOverflowDetect, 106
SetEnableReadoutOccupancy, 106
SetEnableReadoutSeparator, 106
SetEnableRelative, 107
SetEnableResetChannelBufferWhenSeparator, 107
SetEnableSeparatorOnBunchReset, 107
SetEnableSeparatorOnEventReset, 108
SetEnableSerial, 108
SetEnableSetCountersOnBunchReset, 108
SetEnableTTLClock, 109
SetEnableTTLControl, 109
SetEnableTTLHit, 109
SetEnableTTLReset, 110
SetEnableTTLSerial, 110
SetEventCountOffset, 111
SetIOClockDelay, 111
SetIOClockSource, 111
SetKeepToken, 112
SetLeadingMode, 112
SetLowPowerMode, 112
SetMaster, 113
SetMatchWindow, 113
SetMaxEventSize, 113
SetModeRC, 114
SetModeRCCompression, 114
SetPLLControl, 114
SetRCAdjustment, 115
SetReadoutFIFOSize, 115
SetReadoutSingleCycleSpeed, 115
SetReadoutSpeedSelect, 116
SetRejectCountOffset, 116
SetRejectFIFOFull, 116
SetRollOver, 117
SetSearchWindow, 117
SetSerialClockDelay, 117
SetSerialClockSource, 118
SetSerialDelay, 118
SetSetupParity, 118
SetStrobeSelect, 119
SetTestInvert, 119
SetTestMode, 119
SetTokenDelay, 120
SetTrailingMode, 120
SetTriggerCountOffset, 120
SetTriggerMatchingMode, 121
SetVernierOffset, 121
SetWidthResolution, 121
TDCSetup, 82, 83
TDCStatus, 125
kDLLLock, 127
kError, 127
kHaveToken, 127
kL1Occupancy, 127
kReadoutFIFOEmpty, 127

kReadoutFIFOFull, 127
kReadoutFIFOOccupancy, 127
kTriggerFIFOEmpty, 127
kTriggerFIFOFull, 127
kTriggerFIFOOccupancy, 127
SetConstantValues, 126
TDCStatus, 126
TDCTrailer
HPTDC chip control, 11
TrailingEdge
HPTDC chip control, 11
TriggerFIFOParityError
HPTDC chip control, 11
TriggerMatchingError
HPTDC chip control, 11
Type
Exception, 22
TypeString
Exception, 22

USBHandler, 127
∼USBHandler, 128
DumpDevice, 128
fDevice, 129
fHandle, 129
fIsStopping, 129
fProductId, 129
fVendorId, 129
Fetch, 128
Init, 128
Reset, 129
USBHandler, 128
Write, 129

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, 8
WidthResolution
HPTDC chip control, 13
word_t
TDCRegister, 73
Write
USBHandler, 129
WriteRegister
TDC, 57