FlyCapture2 C

2.13.3.31

Generated by Doxygen 1.8.13

Contents

1 Software Licensing Information **1**

2 Deprecated List **3**

3 Module Index **5**

3.1 Modules 5

4 Data Structure Index **7**

4.1 Data Structures 7

5 File Index **9**

5.1 File List 9

6 Module Documentation **11**

6.1 Bus Manager Operation 11

6.1.1 Detailed Description 12

6.1.2 Function Documentation 12

6.1.2.1 fc2DiscoverGigECameras() 12

6.1.2.2 fc2FireBusReset() 13

6.1.2.3 fc2ForceAllIPAddressesAutomatically() 13

6.1.2.4 fc2ForceIPAddressAutomatically() 14

6.1.2.5 fc2ForceIPAddressToCamera() 14

6.1.2.6 fc2GetCameraFromIndex() 14

6.1.2.7 fc2GetCameraFromIPAddress() 16

6.1.2.8 fc2GetCameraFromSerialNumber() 16

6.1.2.9 fc2GetCameraSerialNumberFromIndex() 17

6.1.2.10 fc2GetDeviceFromIndex() 17
6.1.2.11 fc2GetInterfaceTypeFromGuid() 18
6.1.2.12 fc2GetNumOfCameras() 18
6.1.2.13 fc2GetNumOfDevices() 19
6.1.2.14 fc2GetTopology() 19
6.1.2.15 fc2GetUsbLinkInfo() 19
6.1.2.16 fc2GetUsbPortStatus() 20
6.1.2.17 fc2IsCameraControlable() 20
6.1.2.18 fc2ReadPhyRegister() 21
6.1.2.19 fc2RegisterCallback() 21
6.1.2.20 fc2RescanBus() 22
6.1.2.21 fc2UnregisterCallback() 22
6.1.2.22 fc2WritePhyRegister() 22
6.2 Connection and Image Retrieval 25
6.2.1 Detailed Description 25
6.2.2 Function Documentation 25
6.2.2.1 fc2Connect() 26
6.2.2.2 fc2Disconnect() 26
6.2.2.3 fc2GetConfiguration() 26
6.2.2.4 fc2IsConnected() 27
6.2.2.5 fc2RetrieveBuffer() 27
6.2.2.6 fc2SetCallback() 28
6.2.2.7 fc2SetConfiguration() 28
6.2.2.8 fc2SetUserBuffers() 29
6.2.2.9 fc2StartCapture() 29
6.2.2.10 fc2StartCaptureCallback() 30
6.2.2.11 fc2StartSyncCapture() 31
6.2.2.12 fc2StartSyncCaptureCallback() 31
6.2.2.13 fc2StopCapture() 32
6.2.2.14 fc2WaitForBufferEvent() 32

6.3 Information and Properties 34
6.3.1 Detailed Description 34
6.3.2 Function Documentation 34
6.3.2.1 fc2GetCameraInfo() 34
6.3.2.2 fc2GetProperty() 35
6.3.2.3 fc2GetPropertyInfo() 35
6.3.2.4 fc2SetProperty() 36
6.3.2.5 fc2SetPropertyBroadcast() 36
6.4 General Purpose Input / Output 38
6.4.1 Detailed Description 38
6.4.2 Function Documentation 38
6.4.2.1 fc2GetGPIOPinDirection() 38
6.4.2.2 fc2SetGPIOPinDirection() 39
6.4.2.3 fc2SetGPIOPinDirectionBroadcast() 39
6.5 Trigger 41
6.5.1 Detailed Description 41
6.5.2 Function Documentation 41
6.5.2.1 fc2FireSoftwareTrigger() 41
6.5.2.2 fc2FireSoftwareTriggerBroadcast() 42
6.5.2.3 fc2GetTriggerDelay() 42
6.5.2.4 fc2GetTriggerDelayInfo() 43
6.5.2.5 fc2GetTriggerMode() 43
6.5.2.6 fc2GetTriggerModeInfo() 44
6.5.2.7 fc2SetTriggerDelay() 44
6.5.2.8 fc2SetTriggerDelayBroadcast() 45
6.5.2.9 fc2SetTriggerMode() 46
6.5.2.10 fc2SetTriggerModeBroadcast() 46
6.6 Strobe 48
6.6.1 Detailed Description 48
6.6.2 Function Documentation 48

6.6.2.1 fc2GetStrobe() 48
6.6.2.2 fc2GetStrobeInfo() 49
6.6.2.3 fc2SetStrobe() 49
6.6.2.4 fc2SetStrobeBroadcast() 50
6.7 Look Up Table 51
6.7.1 Detailed Description 51
6.7.2 Function Documentation 51
6.7.2.1 fc2EnableLUT() 51
6.7.2.2 fc2GetActiveLUTBank() 52
6.7.2.3 fc2GetLUTBankInfo() 52
6.7.2.4 fc2GetLUTChannel() 53
6.7.2.5 fc2GetLUTInfo() 53
6.7.2.6 fc2SetActiveLUTBank() 54
6.7.2.7 fc2SetLUTChannel() 54
6.8 Memory Channels 56
6.8.1 Detailed Description 56
6.8.2 Function Documentation 56
6.8.2.1 fc2GetEmbeddedImageInfo() 56
6.8.2.2 fc2GetMemoryChannel() 57
6.8.2.3 fc2GetMemoryChannelInfo() 57
6.8.2.4 fc2RestoreFromMemoryChannel() 58
6.8.2.5 fc2SaveToMemoryChannel() 58
6.8.2.6 fc2SetEmbeddedImageInfo() 59
6.9 Register Operation 60
6.9.1 Detailed Description 60
6.9.2 Function Documentation 60
6.9.2.1 fc2GetRegisterString() 60
6.9.2.2 fc2ReadRegister() 61
6.9.2.3 fc2ReadRegisterBlock() 61
6.9.2.4 fc2WriteRegister() 62

6.9.2.5 fc2WriteRegisterBlock() 62
6.9.2.6 fc2WriteRegisterBroadcast() 63
6.10 DCAM Formats 64
6.10.1 Detailed Description 64
6.10.2 Function Documentation 64
6.10.2.1 fc2GetVideoModeAndFrameRate() 64
6.10.2.2 fc2GetVideoModeAndFrameRateInfo() 65
6.10.2.3 fc2SetVideoModeAndFrameRate() 65
6.11 Format7 66
6.11.1 Detailed Description 66
6.11.2 Function Documentation 66
6.11.2.1 fc2GetFormat7Configuration() 66
6.11.2.2 fc2GetFormat7Info() 67
6.11.2.3 fc2SetFormat7Configuration() 67
6.11.2.4 fc2SetFormat7ConfigurationPacket() 68
6.11.2.5 fc2ValidateFormat7Settings() 68
6.12 GVCP Register Operation 69
6.12.1 Detailed Description 69
6.12.2 Function Documentation 69
6.12.2.1 fc2ReadGVCPMemory() 69
6.12.2.2 fc2ReadGVCPRegister() 70
6.12.2.3 fc2ReadGVCPRegisterBlock() 70
6.12.2.4 fc2WriteGVCPMemory() 71
6.12.2.5 fc2WriteGVCPRegister() 71
6.12.2.6 fc2WriteGVCPRegisterBlock() 72
6.12.2.7 fc2WriteGVCPRegisterBroadcast() 72
6.13 GigE property manipulation 73
6.13.1 Detailed Description 73
6.13.2 Function Documentation 73
6.13.2.1 fc2DiscoverGigEPacketSize() 73

6.13.2.2 fc2GetGigEProperty() 74
6.13.2.3 fc2SetGigEProperty() 74
6.14 GigE image settings 75
6.14.1 Detailed Description 75
6.14.2 Function Documentation 75
6.14.2.1 fc2GetGigEImageSettings() 75
6.14.2.2 fc2GetGigEImageSettingsInfo() 76
6.14.2.3 fc2GetGigEImagingMode() 76
6.14.2.4 fc2QueryGigEImagingMode() 76
6.14.2.5 fc2SetGigEImageSettings() 78
6.14.2.6 fc2SetGigEImagingMode() 78
6.15 GigE image binning settings 80
6.15.1 Detailed Description 80
6.15.2 Function Documentation 80
6.15.2.1 fc2GetGigEImageBinningSettings() 80
6.15.2.2 fc2SetGigEImageBinningSettings() 80
6.16 GigE image stream configuration 82
6.16.1 Detailed Description 82
6.16.2 Function Documentation 82
6.16.2.1 fc2GetGigEConfig() 82
6.16.2.2 fc2GetGigEStreamChannelInfo() 83
6.16.2.3 fc2GetNumStreamChannels() 83
6.16.2.4 fc2SetGigEConfig() 83
6.16.2.5 fc2SetGigEStreamChannelInfo() 84
6.17 Image Operation 85
6.17.1 Detailed Description 86
6.17.2 Function Documentation 86
6.17.2.1 fc2CalculateImageStatistics() 86
6.17.2.2 fc2ConvertImage() 86
6.17.2.3 fc2ConvertImageTo() 87

6.17.2.4 fc2CreateImage() . . . 87
6.17.2.5 fc2DestroyImage() . . . 88
6.17.2.6 fc2DetermineBitsPerPixel() . . . 88
6.17.2.7 fc2GetDefaultColorProcessing() . . . 89
6.17.2.8 fc2GetDefaultOutputFormat() . . . 89
6.17.2.9 fc2GetImageColorProcessing() . . . 89
6.17.2.10 fc2GetImageData() . . . 90
6.17.2.11 fc2GetImageDimensions() . . . 90
6.17.2.12 fc2GetImageMetadata() . . . 91
6.17.2.13 fc2GetImageTimeStamp() . . . 91
6.17.2.14 fc2SaveImage() . . . 91
6.17.2.15 fc2SaveImageWithOption() . . . 92
6.17.2.16 fc2SetDefaultColorProcessing() . . . 92
6.17.2.17 fc2SetDefaultOutputFormat() . . . 93
6.17.2.18 fc2SetImageColorProcessing() . . . 93
6.17.2.19 fc2SetImageData() . . . 94
6.17.2.20 fc2SetImageDimensions() . . . 94
6.18 Image Statistics Operation . . . 95
6.18.1 Detailed Description . . . 96
6.18.2 Function Documentation . . . 96
6.18.2.1 fc2CreateImageStatistics() . . . 96
6.18.2.2 fc2DestroyImageStatistics() . . . 96
6.18.2.3 fc2GetChannelHistogram() . . . 96
6.18.2.4 fc2GetChannelMean() . . . 97
6.18.2.5 fc2GetChannelNumPixelValues() . . . 97
6.18.2.6 fc2GetChannelPixelValueRange() . . . 98
6.18.2.7 fc2GetChannelRange() . . . 98
6.18.2.8 fc2GetChannelStatus() . . . 99
6.18.2.9 fc2GetImageStatistics() . . . 99
6.18.2.10 fc2ImageStatisticsDisableAll() . . . 100

6.18.2.11 fc2ImageStatisticsEnableAll() 100
6.18.2.12 fc2ImageStatisticsEnableGreyOnly() 101
6.18.2.13 fc2ImageStatisticsEnableHSLOnly() 101
6.18.2.14 fc2ImageStatisticsEnableRGBOnly() 101
6.18.2.15 fc2SetChannelStatus() 102
6.19 TopologyNode Operation 103
6.19.1 Detailed Description 104
6.19.2 Function Documentation 104
6.19.2.1 fc2CreateTopologyNode() 104
6.19.2.2 fc2DestroyTopologyNode() 104
6.19.2.3 fc2TopologyNodeAddChild() 104
6.19.2.4 fc2TopologyNodeAddPortType() 105
6.19.2.5 fc2TopologyNodeAssignGuidToNode() 105
6.19.2.6 fc2TopologyNodeAssignGuidToNodeEx() 106
6.19.2.7 fc2TopologyNodeGetChild() 106
6.19.2.8 fc2TopologyNodeGetDeviceId() 107
6.19.2.9 fc2TopologyNodeGetGuid() 107
6.19.2.10 fc2TopologyNodeGetInterfaceType() 107
6.19.2.11 fc2TopologyNodeGetNodeType() 108
6.19.2.12 fc2TopologyNodeGetNumChildren() 108
6.19.2.13 fc2TopologyNodeGetNumPorts() 108
6.19.2.14 fc2TopologyNodeGetPortType() 109
6.20 Utilities 110
6.20.1 Detailed Description 110
6.20.2 Function Documentation 110
6.20.2.1 fc2CheckDriver() 110
6.20.2.2 fc2ErrorToDescription() 111
6.20.2.3 fc2GetDriverDeviceName() 111
6.20.2.4 fc2GetLibraryVersion() 112
6.20.2.5 fc2GetSystemInfo() 112

6.20.2.6 fc2LaunchBrowser() 112
6.20.2.7 fc2LaunchCommand() 113
6.20.2.8 fc2LaunchCommandAsync() 113
6.20.2.9 fc2LaunchHelp() 113
6.21 TypeDefs 115
6.21.1 Detailed Description 115
6.21.2 Macro Definition Documentation 115
6.21.2.1 FALSE 115
6.21.2.2 FULL_32BIT_VALUE 115
6.21.2.3 MAX_STRING_LENGTH 116
6.21.2.4 TRUE 116
6.21.3 Typedef Documentation 116
6.21.3.1 BOOL 116
6.21.3.2 fc2Context 116
6.21.3.3 fc2GuiContext 116
6.21.3.4 fc2ImageImpl 116
6.21.3.5 fc2ImageStatisticsContext 117
6.21.3.6 fc2TopologyNodeContext 117
6.21.3.7 fc2VideoContext 117
6.22 Enumerations 118
6.22.1 Detailed Description 122
6.22.2 Enumeration Type Documentation 122
6.22.2.1 fc2BandwidthAllocation 123
6.22.2.2 fc2BayerTileFormat 123
6.22.2.3 fc2BusCallbackType 123
6.22.2.4 fc2BusSpeed 124
6.22.2.5 fc2ColorProcessingAlgorithm 124
6.22.2.6 fc2DriverType 125
6.22.2.7 fc2Error 125
6.22.2.8 fc2FrameRate 126

6.22.2.9 fc2GrabMode . . . 127
6.22.2.10 fc2GrabTimeout . . . 127
6.22.2.11 fc2ImageFileFormat . . . 128
6.22.2.12 fc2InterfaceType . . . 128
6.22.2.13 fc2Mode . . . 129
6.22.2.14 fc2PCIeBusSpeed . . . 129
6.22.2.15 fc2PixelFormat . . . 130
6.22.2.16 fc2PropertyType . . . 130
6.22.2.17 fc2VideoMode . . . 131
6.23 GigE specific enumerations . . . 133
6.23.1 Detailed Description . . . 133
6.23.2 Enumeration Type Documentation . . . 133
6.23.2.1 fc2GigEPropertyType . . . 133
6.24 Structures . . . 134
6.24.1 Detailed Description . . . 135
6.25 GigE specific structures . . . 136
6.25.1 Detailed Description . . . 136
6.26 IIDC specific structures . . . 137
6.26.1 Detailed Description . . . 137
6.27 Image saving structures. . . . 138
6.27.1 Detailed Description . . . 139
6.27.2 Typedef Documentation . . . 139
6.27.2.1 fc2AsyncCommandCallback . . . 139
6.27.2.2 fc2BusEventCallback . . . 139
6.27.2.3 fc2CallbackHandle . . . 139
6.27.2.4 fc2CameraEventCallback . . . 139
6.27.2.5 fc2ImageEventCallback . . . 139
6.27.3 Enumeration Type Documentation . . . 139
6.27.3.1 fc2TIFFCompressionMethod . . . 139
6.28 Video Recording Operation . . . 141
6.28.1 Detailed Description . . . 141
6.28.2 Function Documentation . . . 141
6.28.2.1 fc2VideoAppend() . . . 141
6.28.2.2 fc2VideoAVIOpen() . . . 142
6.28.2.3 fc2VideoClose() . . . 142
6.28.2.4 fc2VideoCreate() . . . 143
6.28.2.5 fc2VideoDestroy() . . . 143
6.28.2.6 fc2VideoH264Open() . . . 143
6.28.2.7 fc2VideoMJPGOpen() . . . 144
6.28.2.8 fc2VideoSetMaximumSize() . . . 144
6.29 Video saving structures. . . . 146
6.29.1 Detailed Description . . . 146

7 Data Structure Documentation 147
7.1 fc2AVIOption Struct Reference 147
7.1.1 Detailed Description 147
7.1.2 Field Documentation 147
7.1.2.1 frameRate 147
7.1.2.2 reserved 148
7.2 fc2BMPOption Struct Reference 148
7.2.1 Detailed Description 148
7.2.2 Field Documentation 148
7.2.2.1 indexedColor_8bit 148
7.2.2.2 reserved 148
7.3 fc2CameraInfo Struct Reference 149
7.3.1 Detailed Description 150
7.3.2 Field Documentation 150
7.3.2.1 applicationIPAddress 151
7.3.2.2 applicationPort 151
7.3.2.3 bayerTileFormat 151
7.3.2.4 busNumber 151
7.3.2.5 ccpStatus 151
7.3.2.6 configROM 151
7.3.2.7 defaultGateway 152
7.3.2.8 driverName 152
7.3.2.9 driverType 152
7.3.2.10 firmwareBuildTime 152
7.3.2.11 firmwareVersion 152
7.3.2.12 gigEMajorVersion 152
7.3.2.13 gigEMinorVersion 153
7.3.2.14 iidcVer 153
7.3.2.15 interfaceType 153
7.3.2.16 ipAddress 153

7.3.2.17 isColorCamera 153
7.3.2.18 macAddress 153
7.3.2.19 maximumBusSpeed 154
7.3.2.20 modelName 154
7.3.2.21 nodeNumber 154
7.3.2.22 pcieBusSpeed 154
7.3.2.23 reserved 154
7.3.2.24 sensorInfo 154
7.3.2.25 sensorResolution 155
7.3.2.26 serialNumber 155
7.3.2.27 subnetMask 155
7.3.2.28 userDefinedName 155
7.3.2.29 vendorName 155
7.3.2.30 xmlURL1 155
7.3.2.31 xmlURL2 156
7.4 fc2CameraStats Struct Reference 156
7.4.1 Detailed Description 157
7.4.2 Field Documentation 157
7.4.2.1 cameraCurrents 157
7.4.2.2 cameraPowerUp 157
7.4.2.3 cameraVoltages 157
7.4.2.4 imageCorrupt 157
7.4.2.5 imageDriverDropped 157
7.4.2.6 imageDropped 157
7.4.2.7 imageXmitFailed 158
7.4.2.8 numCurrents 158
7.4.2.9 numResendPacketsReceived 158
7.4.2.10 numResendPacketsRequested 158
7.4.2.11 numVoltages 158
7.4.2.12 portErrors 158

7.4.2.13 regReadFailed . . . 158
7.4.2.14 regWriteFailed . . . 159
7.4.2.15 reserved . . . 159
7.4.2.16 temperature . . . 159
7.4.2.17 timeSinceBusReset . . . 159
7.4.2.18 timeSinceInitialization . . . 159
7.4.2.19 timeStamp . . . 159
7.5 fc2Config Struct Reference . . . 159
7.5.1 Detailed Description . . . 160
7.5.2 Field Documentation . . . 160
7.5.2.1 asyncBusSpeed . . . 160
7.5.2.2 bandwidthAllocation . . . 161
7.5.2.3 grabMode . . . 161
7.5.2.4 grabTimeout . . . 161
7.5.2.5 highPerformanceRetrieveBuffer . . . 161
7.5.2.6 isochBusSpeed . . . 161
7.5.2.7 minNumImageNotifications . . . 162
7.5.2.8 numBuffers . . . 162
7.5.2.9 numImageNotifications . . . 162
7.5.2.10 registerTimeout . . . 162
7.5.2.11 registerTimeoutRetries . . . 163
7.5.2.12 reserved . . . 163
7.6 fc2ConfigROM Struct Reference . . . 163
7.6.1 Detailed Description . . . 164
7.6.2 Field Documentation . . . 164
7.6.2.1 chipIdHi . . . 164
7.6.2.2 chipIdLo . . . 164
7.6.2.3 nodeVendorId . . . 164
7.6.2.4 pszKeyword . . . 164
7.6.2.5 reserved . . . 164

7.6.2.6 unitSpecId . . . 165
7.6.2.7 unitSubSWVer . . . 165
7.6.2.8 unitSWVer . . . 165
7.6.2.9 vendorUniqueInfo_0 . . . 165
7.6.2.10 vendorUniqueInfo_1 . . . 165
7.6.2.11 vendorUniqueInfo_2 . . . 165
7.6.2.12 vendorUniqueInfo_3 . . . 166
7.7 fc2EmbeddedImageInfo Struct Reference . . . 166
7.7.1 Detailed Description . . . 167
7.7.2 Field Documentation . . . 167
7.7.2.1 brightness . . . 167
7.7.2.2 exposure . . . 167
7.7.2.3 frameCounter . . . 167
7.7.2.4 gain . . . 167
7.7.2.5 GPIOPinState . . . 167
7.7.2.6 ROIPosition . . . 167
7.7.2.7 shutter . . . 168
7.7.2.8 strobePattern . . . 168
7.7.2.9 timestamp . . . 168
7.7.2.10 whiteBalance . . . 168
7.8 fc2EmbeddedImageInfoProperty Struct Reference . . . 168
7.8.1 Detailed Description . . . 168
7.8.2 Field Documentation . . . 168
7.8.2.1 available . . . 169
7.8.2.2 onOff . . . 169
7.9 fc2EventCallbackData Struct Reference . . . 169
7.9.1 Field Documentation . . . 169
7.9.1.1 EventData . . . 169
7.9.1.2 EventDataSize . . . 170
7.9.1.3 EventID . . . 170

7.9.1.4 EventName 170
7.9.1.5 EventTimestamp 170
7.9.1.6 EventUserData 170
7.9.1.7 EventUserDataSize 170
7.10 fc2EventOptions Struct Reference 171
7.10.1 Detailed Description 171
7.10.2 Field Documentation 171
7.10.2.1 EventCallbackFcn 171
7.10.2.2 EventName 171
7.10.2.3 EventUserData 171
7.10.2.4 EventUserDataSize 172
7.11 fc2Format7ImageSettings Struct Reference 172
7.11.1 Detailed Description 172
7.11.2 Field Documentation 172
7.11.2.1 height 172
7.11.2.2 mode 173
7.11.2.3 offsetX 173
7.11.2.4 offsetY 173
7.11.2.5 pixelFormat 173
7.11.2.6 reserved 173
7.11.2.7 width 173
7.12 fc2Format7Info Struct Reference 174
7.12.1 Detailed Description 174
7.12.2 Field Documentation 174
7.12.2.1 imageHStepSize 174
7.12.2.2 imageVStepSize 175
7.12.2.3 maxHeight 175
7.12.2.4 maxPacketSize 175
7.12.2.5 maxWidth 175
7.12.2.6 minPacketSize 175

7.12.2.7 mode 175
7.12.2.8 offsetHStepSize 176
7.12.2.9 offsetVStepSize 176
7.12.2.10 packetSize 176
7.12.2.11 percentage 176
7.12.2.12 pixelFormatBitField 176
7.12.2.13 reserved 176
7.12.2.14 vendorPixelFormatBitField 177
7.13 fc2Format7PacketInfo Struct Reference 177
7.13.1 Detailed Description 177
7.13.2 Field Documentation 177
7.13.2.1 maxBytesPerPacket 177
7.13.2.2 recommendedBytesPerPacket 178
7.13.2.3 reserved 178
7.13.2.4 unitBytesPerPacket 178
7.14 fc2GigEConfig Struct Reference 178
7.14.1 Detailed Description 178
7.14.2 Field Documentation 179
7.14.2.1 enablePacketResend 179
7.14.2.2 registerTimeout 179
7.14.2.3 registerTimeoutRetries 179
7.14.2.4 reserved 179
7.15 fc2GigEImageSettings Struct Reference 179
7.15.1 Detailed Description 180
7.15.2 Field Documentation 180
7.15.2.1 height 180
7.15.2.2 offsetX 180
7.15.2.3 offsetY 180
7.15.2.4 pixelFormat 181
7.15.2.5 reserved 181

7.15.2.6 width 181
7.16 fc2GigEImageSettingsInfo Struct Reference 181
7.16.1 Detailed Description 182
7.16.2 Field Documentation 182
7.16.2.1 imageHStepSize 182
7.16.2.2 imageVStepSize 182
7.16.2.3 maxHeight 182
7.16.2.4 maxWidth 182
7.16.2.5 offsetHStepSize 182
7.16.2.6 offsetVStepSize 183
7.16.2.7 pixelFormatBitField 183
7.16.2.8 reserved 183
7.16.2.9 vendorPixelFormatBitField 183
7.17 fc2GigEProperty Struct Reference 183
7.17.1 Detailed Description 184
7.17.2 Field Documentation 184
7.17.2.1 isReadable 184
7.17.2.2 isWritable 184
7.17.2.3 max 184
7.17.2.4 min 184
7.17.2.5 propType 184
7.17.2.6 reserved 185
7.17.2.7 value 185
7.18 fc2GigEStreamChannel Struct Reference 185
7.18.1 Detailed Description 186
7.18.2 Field Documentation 186
7.18.2.1 destinationIpAddress 186
7.18.2.2 doNotFragment 186
7.18.2.3 hostPort 186
7.18.2.4 interPacketDelay 186

7.18.2.5 networkInterfaceIndex 186
7.18.2.6 packetSize 187
7.18.2.7 reserved 187
7.18.2.8 sourcePort 187
7.19 fc2H264Option Struct Reference 187
7.19.1 Detailed Description 187
7.19.2 Field Documentation 188
7.19.2.1 bitrate 188
7.19.2.2 frameRate 188
7.19.2.3 height 188
7.19.2.4 reserved 188
7.19.2.5 width 188
7.20 fc2Image Struct Reference 189
7.20.1 Field Documentation 189
7.20.1.1 bayerFormat 189
7.20.1.2 cols 189
7.20.1.3 dataSize 189
7.20.1.4 format 189
7.20.1.5 imageImpl 189
7.20.1.6 pData 190
7.20.1.7 receivedDataSize 190
7.20.1.8 rows 190
7.20.1.9 stride 190
7.21 fc2ImageMetadata Struct Reference 190
7.21.1 Detailed Description 191
7.21.2 Field Documentation 191
7.21.2.1 embeddedBrightness 191
7.21.2.2 embeddedExposure 191
7.21.2.3 embeddedFrameCounter 191
7.21.2.4 embeddedGain 191

7.21.2.5 embeddedGPIOPinState 191
7.21.2.6 embeddedROIPosition 192
7.21.2.7 embeddedShutter 192
7.21.2.8 embeddedStrobePattern 192
7.21.2.9 embeddedTimeStamp 192
7.21.2.10 embeddedWhiteBalance 192
7.21.2.11 reserved 192
7.22 fc2InternalContext Struct Reference 193
7.22.1 Field Documentation 193
7.22.1.1 pBusMgr 193
7.22.1.2 pCamera 193
7.23 fc2InternalGuiContext Struct Reference 193
7.23.1 Field Documentation 193
7.23.1.1 pCameraControlDlg 193
7.23.1.2 pCameraSelectionDlg 194
7.24 fc2InternalImageCallback Struct Reference 194
7.24.1 Field Documentation 194
7.24.1.1 pCallback 194
7.24.1.2 pCallbackData 194
7.25 fc2IPAddress Struct Reference 195
7.25.1 Detailed Description 195
7.25.2 Field Documentation 195
7.25.2.1 octets 195
7.26 fc2JPEGOption Struct Reference 195
7.26.1 Detailed Description 195
7.26.2 Field Documentation 196
7.26.2.1 progressive 196
7.26.2.2 quality 196
7.26.2.3 reserved 196
7.27 fc2JPG2Option Struct Reference 196

7.27.1 Detailed Description 197
7.27.2 Field Documentation 197
7.27.2.1 quality 197
7.27.2.2 reserved 197
7.28 fc2LUTData Struct Reference 197
7.28.1 Detailed Description 198
7.28.2 Field Documentation 198
7.28.2.1 enabled 198
7.28.2.2 inputBitDepth 198
7.28.2.3 numBanks 198
7.28.2.4 numChannels 198
7.28.2.5 numEntries 198
7.28.2.6 outputBitDepth 199
7.28.2.7 reserved 199
7.28.2.8 supported 199
7.29 fc2MACAddress Struct Reference 199
7.29.1 Detailed Description 199
7.29.2 Field Documentation 199
7.29.2.1 octets 200
7.30 fc2MJPGOption Struct Reference 200
7.30.1 Detailed Description 200
7.30.2 Field Documentation 200
7.30.2.1 frameRate 200
7.30.2.2 quality 200
7.30.2.3 reserved 201
7.31 fc2PGMOption Struct Reference 201
7.31.1 Detailed Description 201
7.31.2 Field Documentation 201
7.31.2.1 binaryFile 201
7.31.2.2 reserved 201

7.32 fc2PGRGuid Struct Reference 202
7.32.1 Detailed Description 202
7.32.2 Field Documentation 202
7.32.2.1 value 202
7.33 fc2PNGOption Struct Reference 202
7.33.1 Detailed Description 202
7.33.2 Field Documentation 203
7.33.2.1 compressionLevel 203
7.33.2.2 interlaced 203
7.33.2.3 reserved 203
7.34 fc2PPMOption Struct Reference 203
7.34.1 Detailed Description 203
7.34.2 Field Documentation 204
7.34.2.1 binaryFile 204
7.34.2.2 reserved 204
7.35 fc2StrobeControl Struct Reference 204
7.35.1 Detailed Description 204
7.35.2 Field Documentation 205
7.35.2.1 delay 205
7.35.2.2 duration 205
7.35.2.3 onOff 205
7.35.2.4 polarity 205
7.35.2.5 reserved 205
7.35.2.6 source 206
7.36 fc2StrobeInfo Struct Reference 206
7.36.1 Detailed Description 206
7.36.2 Field Documentation 206
7.36.2.1 maxValue 206
7.36.2.2 minValue 207
7.36.2.3 onOffSupported 207

7.36.2.4 polaritySupported . . . 207
7.36.2.5 present . . . 207
7.36.2.6 readOutSupported . . . 207
7.36.2.7 reserved . . . 207
7.36.2.8 source . . . 208
7.37 fc2SystemInfo Struct Reference . . . 208
7.37.1 Detailed Description . . . 208
7.37.2 Field Documentation . . . 209
7.37.2.1 byteOrder . . . 209
7.37.2.2 cpuDescription . . . 209
7.37.2.3 driverList . . . 209
7.37.2.4 gpuDescription . . . 209
7.37.2.5 libraryList . . . 209
7.37.2.6 numCpuCores . . . 210
7.37.2.7 osDescription . . . 210
7.37.2.8 osType . . . 210
7.37.2.9 reserved . . . 210
7.37.2.10 screenHeight . . . 210
7.37.2.11 screenWidth . . . 210
7.37.2.12 sysMemSize . . . 211
7.38 fc2TIFFOption Struct Reference . . . 211
7.38.1 Detailed Description . . . 211
7.38.2 Field Documentation . . . 211
7.38.2.1 compression . . . 211
7.38.2.2 reserved . . . 211
7.39 fc2TimeStamp Struct Reference . . . 212
7.39.1 Detailed Description . . . 212
7.39.2 Field Documentation . . . 212
7.39.2.1 cycleCount . . . 212
7.39.2.2 cycleOffset . . . 212

7.39.2.3 cycleSeconds 213
7.39.2.4 microSeconds 213
7.39.2.5 reserved 213
7.39.2.6 seconds 213
7.40 fc2TriggerDelay Struct Reference 213
7.40.1 Detailed Description 214
7.40.2 Field Documentation 214
7.40.2.1 absControl 214
7.40.2.2 absValue 215
7.40.2.3 autoManualMode 215
7.40.2.4 onePush 215
7.40.2.5 onOff 215
7.40.2.6 present 215
7.40.2.7 reserved 215
7.40.2.8 type 216
7.40.2.9 valueA 216
7.40.2.10 valueB 216
7.41 fc2TriggerDelayInfo Struct Reference 216
7.41.1 Detailed Description 217
7.41.2 Field Documentation 217
7.41.2.1 absMax 217
7.41.2.2 absMin 217
7.41.2.3 absValSupported 217
7.41.2.4 autoSupported 218
7.41.2.5 manualSupported 218
7.41.2.6 max 218
7.41.2.7 min 218
7.41.2.8 onePushSupported 218
7.41.2.9 onOffSupported 218
7.41.2.10 present 219

7.41.2.11 pUnitAbbr 219
7.41.2.12 pUnits 219
7.41.2.13 readOutSupported 219
7.41.2.14 reserved 219
7.41.2.15 type 219
7.42 fc2TriggerMode Struct Reference 220
7.42.1 Detailed Description 220
7.42.2 Field Documentation 220
7.42.2.1 mode 220
7.42.2.2 onOff 220
7.42.2.3 parameter 221
7.42.2.4 polarity 221
7.42.2.5 reserved 221
7.42.2.6 source 221
7.43 fc2TriggerModeInfo Struct Reference 221
7.43.1 Detailed Description 222
7.43.2 Field Documentation 222
7.43.2.1 modeMask 222
7.43.2.2 onOffSupported 222
7.43.2.3 polaritySupported 223
7.43.2.4 present 223
7.43.2.5 readOutSupported 223
7.43.2.6 reserved 223
7.43.2.7 softwareTriggerSupported 223
7.43.2.8 sourceMask 223
7.43.2.9 valueReadable 224
7.44 fc2Version Struct Reference 224
7.44.1 Detailed Description 224
7.44.2 Field Documentation 224
7.44.2.1 build 224
7.44.2.2 major 225
7.44.2.3 minor 225
7.44.2.4 type 225

8 File Documentation 227

8.1 FlyCapture2_C.h File Reference . . . 227

8.1.1 Function Documentation . . . 236

8.1.1.1 fc2CreateContext() . . . 236

8.1.1.2 fc2CreateGigEContext() . . . 237

8.1.1.3 fc2DeregisterAllEvents() . . . 237

8.1.1.4 fc2DeregisterEvent() . . . 237

8.1.1.5 fc2DestroyContext() . . . 238

8.1.1.6 fc2GetCycleTime() . . . 238

8.1.1.7 fc2GetStats() . . . 239

8.1.1.8 fc2RegisterAllEvents() . . . 239

8.1.1.9 fc2RegisterEvent() . . . 240

8.1.1.10 ResetStats() . . . 240

8.2 FlyCapture2Defs_C.h File Reference . . . 240

8.2.1 Enumeration Type Documentation . . . 248

8.2.1.1 fc2ByteOrder . . . 248

8.2.1.2 fc2NodeType . . . 249

8.2.1.3 fc2OSType . . . 249

8.2.1.4 fc2PortType . . . 249

8.2.1.5 fc2StatisticsChannel . . . 250

8.3 FlyCapture2GUI_C.h File Reference . . . 250

8.3.1 Function Documentation . . . 251

8.3.1.1 fc2CreateGUIContext() . . . 251

8.3.1.2 fc2DestroyGUIContext() . . . 251

8.3.1.3 fc2Disonnect() . . . 252

8.3.1.4 fc2GUIConnect() . . . 252

8.3.1.5 fc2GUIDisconnect() . . . 253

8.3.1.6 fc2Hide() . . . 253

8.3.1.7 fc2IsVisible() . . . 253

8.3.1.8 fc2Show() . . . 254

8.3.1.9 fc2ShowModal() 254
8.4 FlyCapture2Internal_C.h File Reference 255
8.4.1 Function Documentation 255
8.4.1.1 IsContextValid() 255
8.4.1.2 IsGuiContextValid() 255
8.4.1.3 SyncCppImageToStruct() 256
8.5 FlyCapture2Platform_C.h File Reference 256
8.5.1 Macro Definition Documentation 256
8.5.1.1 FLYCAPTURE2_C_API 256
8.5.1.2 FLYCAPTURE2_C_CALL_CONVEN 256
8.6 FlyCapture2Private_C.h File Reference 257
8.6.1 Function Documentation 257
8.6.1.1 GetInternal() 257
8.7 FlyCapture2Video_C.h File Reference 257
8.8 FlyCapture2VideoDefs_C.h File Reference 258
8.9 Licensing.dox File Reference 259
8.10 MultiSyncLibrary_C.h File Reference 259
8.10.1 Function Documentation 260
8.10.1.1 syncCreateContext() 260
8.10.1.2 syncDestroyContext() 260
8.10.1.3 syncDisableCrossPCSynchronization() 260
8.10.1.4 syncEnableCrossPCSynchronization() 261
8.10.1.5 syncGetStatus() 261
8.10.1.6 syncGetTimeSinceSynced() 261
8.10.1.7 syncIsTimingBusConnected() 263
8.10.1.8 syncQueryCrossPCSynchronizationSetting() 263
8.10.1.9 syncRescanMasterTimingBus() 263
8.10.1.10 syncStart() 265
8.10.1.11 syncStop() 265
8.11 MultiSyncLibraryDefs_C.h File Reference 266

8.11.1 Macro Definition Documentation 267
8.11.1.1 FALSE 267
8.11.1.2 FULL_32BIT_VALUE 267
8.11.1.3 MAX_STRING_LENGTH 267
8.11.1.4 TRUE 267
8.11.2 Typedef Documentation 268
8.11.2.1 BOOL 268
8.11.2.2 syncContext 268
8.11.3 Enumeration Type Documentation 268
8.11.3.1 syncError 268
8.11.3.2 syncMessage 268
8.12 MultiSyncLibraryPlatform_C.h File Reference 269
8.12.1 Macro Definition Documentation 269
8.12.1.1 MULTISYNCLIBRARY_C_API 269
8.12.1.2 MULTISYNCLIBRARY_C_CALL_CONVEN 269

Index **271**

Chapter 1

Software Licensing Information

Table 1.1 License table

Component	License
FlyCapture2	Copyright © 2017 FLIR Integrated Imaging Solutions, Inc. All Rights Reserved. This software is the confidential and proprietary information of FLIR Integrated Imaging Solutions, Inc. ("Confidential Information"). You shall not disclose such Confidential Information and shall use it only in accordance with the terms of the license agreement you entered into with FLIR Integrated Imaging Solutions, Inc. (FLIR). FLIR MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THE SOFTWARE, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. FLIR SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.
AdapterList	The Code Project Open License (CPOL) http://www.codeproject.com/info/cpol10.aspx
Boost	Boost Software License http://www.boost.org/users/license.html
FFMPEG	LGPv2.1 License https://www.ffmpeg.org/legal.html
FreeImage	FreeImage public license http://freeimage.sourceforge.net/freeimage-license.txt
GTK	LGPv2.1 License http://www.gnu.org/licenses/old-licenses/lgpl-2.1.txt
Libusb	LGPLv2.1 License http://www.gnu.org/licenses/old-licenses/lgpl-2.1.txt
Libraw1394	LGPLv2.0 License http://www.gnu.org/licenses/old-licenses/lgpl-2.0.txt

Chapter 2

Deprecated List

Global fc2Disonnect (fc2GuiContext context) __attribute__((deprecated))

This method is deprecated and will be removed in a future FlyCapture2 release. Please use fc2GUIDisconnect instead.

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Bus Manager Operation 11
Connection and Image Retrieval 25
Information and Properties 34
General Purpose Input / Output 38
Trigger 41
Strobe 48
Look Up Table 51
Memory Channels 56
Register Operation 60
DCAM Formats 64
Format7 66
GVCP Register Operation 69
GigE property manipulation 73
GigE image settings 75
GigE image binning settings 80
GigE image stream configuration 82
Image Operation 85
Image Statistics Operation 95
TopologyNode Operation 103
Utilities 110
TypeDefs 115
Enumerations 118
GigE specific enumerations 133
Structures 134
    GigE specific structures 136
    IIDC specific structures 137
    Image saving structures. 138
    Video saving structures. 146
Video Recording Operation 141

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

fc2AVIOption
Options for saving AVI files 147
fc2BMPOption
Options for saving Bitmap image 148
fc2CameraInfo
Camera information 149
fc2CameraStats
Camera diagnostic information 156
fc2Config
Configuration for a camera 159
fc2ConfigROM
Camera configuration ROM 163
fc2EmbeddedImageInfo
Properties of the possible embedded image information 166
fc2EmbeddedImageInfoProperty
Properties of a single embedded image info property 168
fc2EventCallbackData 169
fc2EventOptions
Options for enabling device event registration 171
fc2Format7ImageSettings
Format 7 image settings 172
fc2Format7Info
Format 7 information for a single mode 174
fc2Format7PacketInfo
Format 7 packet information 177
fc2GigEConfig
Configuration for a GigE camera 178
fc2GigEImageSettings
Image settings for a GigE camera 179
fc2GigEImageSettingsInfo
Format 7 information for a single mode 181
fc2GigEProperty
A GigE property 183
fc2GigEStreamChannel
Information about a single GigE stream channel 185

fc2H264Option
Options for saving H264 files . . . 187
fc2Image . . . 189
fc2ImageMetadata
Metadata related to an image . . . 190
fc2InternalContext . . . 193
fc2InternalGuiContext . . . 193
fc2InternalImageCallback . . . 194
fc2IPAddress
IPv4 address . . . 195
fc2JPEGOption
Options for saving JPEG image . . . 195
fc2JPG2Option
Options for saving JPEG2000 image . . . 196
fc2LUTData
Information about the camera's look up table . . . 197
fc2MACAddress
MAC address . . . 199
fc2MJPGOption
Options for saving MJPG files . . . 200
fc2PGMOption
Options for saving PGM images . . . 201
fc2PGRGuid
A GUID to the camera . . . 202
fc2PNGOption
Options for saving PNG images . . . 202
fc2PPMOption
Options for saving PPM images . . . 203
fc2StrobeControl
A camera strobe . . . 204
fc2StrobeInfo
A camera strobe property . . . 206
fc2SystemInfo
Description of the system . . . 208
fc2TIFFOption
Options for saving TIFF images . . . 211
fc2TimeStamp
Timestamp information . . . 212
fc2TriggerDelay
A specific camera property . . . 213
fc2TriggerDelayInfo
Information about a specific camera property . . . 216
fc2TriggerMode
A camera trigger . . . 220
fc2TriggerModeInfo
Information about a camera trigger property . . . 221
fc2Version
The current version of the library . . . 224

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

FlyCapture2_C.h 227
FlyCapture2Defs_C.h 240
FlyCapture2GUI_C.h 250
FlyCapture2Internal_C.h 255
FlyCapture2Platform_C.h 256
FlyCapture2Private_C.h 257
FlyCapture2Video_C.h 257
FlyCapture2VideoDefs_C.h 258
MultiSyncLibrary_C.h 259
MultiSyncLibraryDefs_C.h 266
MultiSyncLibraryPlatform_C.h 269

Chapter 6

Module Documentation

6.1 Bus Manager Operation

The functions in this section provide access to BusManager operations.

Functions

- FLYCAPTURE2_C_API fc2Error fc2FireBusReset (fc2Context context, fc2PGRGuid ∗pGuid)

  Fire a bus reset.
- FLYCAPTURE2_C_API fc2Error fc2GetNumOfCameras (fc2Context context, unsigned int ∗pNumCameras)

  Gets the number of cameras attached to the PC.
- FLYCAPTURE2_C_API fc2Error fc2GetCameraFromIPAddress (fc2Context context, fc2IPAddress ip←
Address, fc2PGRGuid ∗pGuid)

  Gets the PGRGuid for a camera with the specified IPv4 address.
- FLYCAPTURE2_C_API fc2Error fc2GetCameraFromIndex (fc2Context context, unsigned int index, fc2PG←
RGuid ∗pGuid)

  Gets the PGRGuid for a camera on the PC.
- FLYCAPTURE2_C_API fc2Error fc2GetCameraFromSerialNumber (fc2Context context, unsigned int serial←
Number, fc2PGRGuid ∗pGuid)

  Gets the PGRGuid for a camera on the PC.
- FLYCAPTURE2_C_API fc2Error fc2GetCameraSerialNumberFromIndex (fc2Context context, unsigned int index, unsigned int ∗pSerialNumber)

  Gets the serial number of the camera with the specified index.
- FLYCAPTURE2_C_API fc2Error fc2GetInterfaceTypeFromGuid (fc2Context context, fc2PGRGuid ∗pGuid, fc2InterfaceType ∗pInterfaceType)

  Gets the interface type associated with a PGRGuid.
- FLYCAPTURE2_C_API fc2Error fc2GetNumOfDevices (fc2Context context, unsigned int ∗pNumDevices)

  Gets the number of devices.
- FLYCAPTURE2_C_API fc2Error fc2GetDeviceFromIndex (fc2Context context, unsigned int index, fc2PGR←
Guid ∗pGuid)

  Gets the PGRGuid for a device.
- FLYCAPTURE2_C_API fc2Error fc2ReadPhyRegister (fc2Context context, fc2PGRGuid guid, unsigned int page, unsigned int port, unsigned int address, unsigned int ∗pValue)

  Read a phy register on the specified device.
- FLYCAPTURE2_C_API fc2Error fc2WritePhyRegister (fc2Context context, fc2PGRGuid guid, unsigned int page, unsigned int port, unsigned int address, unsigned int value)

Write a phy register on the specified device.
- FLYCAPTURE2_C_API fc2Error fc2GetUsbLinkInfo (fc2Context context, fc2PGRGuid guid, unsigned int ∗pValue)

  Read usb link info for the port that the specified device is connected to.
- FLYCAPTURE2_C_API fc2Error fc2GetUsbPortStatus (fc2Context context, fc2PGRGuid guid, unsigned int ∗pValue)

  Read usb port status for the port that the specified device is connected to.
- FLYCAPTURE2_C_API fc2Error fc2GetTopology (fc2Context context, fc2TopologyNodeContext ∗p←TopologyNodeContext)

  Gets the topology information for the PC.
- FLYCAPTURE2_C_API fc2Error fc2RegisterCallback (fc2Context context, fc2BusEventCallback enum←Callback, fc2BusCallbackType callbackType, void ∗pParameter, fc2CallbackHandle ∗pCallbackHandle)

  Register a callback function that will be called when the specified callback event occurs.
- FLYCAPTURE2_C_API fc2Error fc2UnregisterCallback (fc2Context context, fc2CallbackHandle callback←Handle)

  Unregister a callback function.
- FLYCAPTURE2_C_API fc2Error fc2RescanBus (fc2Context context)

  Force a rescan of the buses.
- FLYCAPTURE2_C_API fc2Error fc2ForceIPAddressToCamera (fc2Context context, fc2MACAddress mac←Address, fc2IPAddress ipAddress, fc2IPAddress subnetMask, fc2IPAddress defaultGateway)

  Force the camera with the specific MAC address to the specified IP address, subnet mask and default gateway.
- FLYCAPTURE2_C_API fc2Error fc2ForceAllIPAddressesAutomatically ()

  Force all cameras on the network to be assigned sequential IP addresses on the same subnet as the network adapters that they are connected to.
- FLYCAPTURE2_C_API fc2Error fc2ForceIPAddressAutomatically (unsigned int serialNumber)

  Force cameras on the network to be assigned sequential IP addresses on the same subnet as the network adapters that it is connected to.
- FLYCAPTURE2_C_API fc2Error fc2DiscoverGigECameras (fc2Context context, fc2CameraInfo ∗gigE←Cameras, unsigned int ∗arraySize)

  Discover all cameras connected to the network even if they reside on a different subnet.
- FLYCAPTURE2_C_API fc2Error fc2IsCameraControlable (fc2Context context, fc2PGRGuid ∗pGuid, BOOL ∗pControlable)

  Query whether a GigE camera is controllable.

6.1.1 Detailed Description

The functions in this section provide access to BusManager operations.

6.1.2 Function Documentation

6.1.2.1 fc2DiscoverGigECameras()

```
FLYCAPTURE2_C_API fc2Error fc2DiscoverGigECameras (
            fc2Context context,
            fc2CameraInfo * gigECameras,
            unsigned int * arraySize )
```

Discover all cameras connected to the network even if they reside on a different subnet.

This is useful in situations where a GigE camera is using Persistent IP and the application's subnet is different from the device subnet. After discovering the camera, it is easy to use ForceIPAddressToCamera() to set a different IP configuration.

Parameters

context	The fc2Context to be used.
gigECameras	Pointer to an array of CameraInfo structures.
arraySize	Size of the array. Number of discovered cameras is returned in the same value.

Returns

An Error indicating the success or failure of the function. If the error is PGRERROR_BUFFER_TOO_SMALL then arraySize will contain the minimum size needed for gigECameras array.

6.1.2.2 fc2FireBusReset()

```
FLYCAPTURE2_C_API fc2Error fc2FireBusReset (
             fc2Context context,
             fc2PGRGuid * pGuid )
```

Fire a bus reset.

The actual bus reset is only fired for the specified 1394 bus, but it will effectively cause a global bus reset for the library.

Parameters

context	The fc2Context to be used.
pGuid	PGRGuid of the camera or the device to cause bus reset.

Returns

An Error indicating the success or failure of the function.

6.1.2.3 fc2ForceAllIPAddressesAutomatically()

```
FLYCAPTURE2_C_API fc2Error fc2ForceAllIPAddressesAutomatically ( )
```

Force all cameras on the network to be assigned sequential IP addresses on the same subnet as the network adapters that they are connected to.

This is useful in situations where GigE Vision cameras are using IP addresses in a subnet different from the host's subnet.

Returns

An Error indicating the success or failure of the function.

6.1.2.4 fc2ForceIPAddressAutomatically()

```
FLYCAPTURE2_C_API fc2Error fc2ForceIPAddressAutomatically (
            unsigned int serialNumber )
```

Force cameras on the network to be assigned sequential IP addresses on the same subnet as the network adapters that it is connected to.

This is useful in situations where GigE Vision cameras are using IP addresses in a subnet different from the host's subnet.

Returns

An Error indicating the success or failure of the function.

6.1.2.5 fc2ForceIPAddressToCamera()

```
FLYCAPTURE2_C_API fc2Error fc2ForceIPAddressToCamera (
            fc2Context context,
            fc2MACAddress macAddress,
            fc2IPAddress ipAddress,
            fc2IPAddress subnetMask,
            fc2IPAddress defaultGateway )
```

Force the camera with the specific MAC address to the specified IP address, subnet mask and default gateway.

This is useful in situations where a GigE Vision camera is using Persistent IP and the application's subnet is different from the device subnet.

Parameters

context	The fc2Context to be used.
macAddress	MAC address of the camera.
ipAddress	IP address to set on the camera.
subnetMask	Subnet mask to set on the camera.
defaultGateway	Default gateway to set on the camera.

Returns

An Error indicating the success or failure of the function.

6.1.2.6 fc2GetCameraFromIndex()

```
FLYCAPTURE2_C_API fc2Error fc2GetCameraFromIndex (
            fc2Context context,
            unsigned int index,
            fc2PGRGuid * pGuid )
```

Gets the PGRGuid for a camera on the PC.

It uniquely identifies the camera specified by the index and is used to identify the camera during a fc2Connect() call.

Parameters

context	The fc2Context to be used.
index	Zero based index of camera.
pGuid	Unique PGRGuid for the camera.

Returns

A fc2Error indicating the success or failure of the function.

6.1.2.7 fc2GetCameraFromIPAddress()

```
FLYCAPTURE2_C_API fc2Error fc2GetCameraFromIPAddress (
             fc2Context context,
             fc2IPAddress ipAddress,
             fc2PGRGuid * pGuid )
```

Gets the PGRGuid for a camera with the specified IPv4 address.

Parameters

context	The fc2Context to be used.
ipAddress	IP address to get GUID for.
pGuid	Unique PGRGuid for the camera.

Returns

A fc2Error indicating the success or failure of the function.

6.1.2.8 fc2GetCameraFromSerialNumber()

```
FLYCAPTURE2_C_API fc2Error fc2GetCameraFromSerialNumber (
             fc2Context context,
             unsigned int serialNumber,
             fc2PGRGuid * pGuid )
```

Gets the PGRGuid for a camera on the PC.

It uniquely identifies the camera specified by the serial number and is used to identify the camera during a fc2↩Connect() call.

Parameters

context	The fc2Context to be used.
serialNumber	Serial number of camera.
pGuid	Unique PGRGuid for the camera.

Returns

A fc2Error indicating the success or failure of the function.

6.1.2.9 fc2GetCameraSerialNumberFromIndex()

```
FLYCAPTURE2_C_API fc2Error fc2GetCameraSerialNumberFromIndex (
            fc2Context context,
            unsigned int index,
            unsigned int * pSerialNumber )
```

Gets the serial number of the camera with the specified index.

Parameters

context	The fc2Context to be used.
index	Zero based index of desired camera.
pSerialNumber	Serial number of camera.

Returns

A fc2Error indicating the success or failure of the function.

6.1.2.10 fc2GetDeviceFromIndex()

```
FLYCAPTURE2_C_API fc2Error fc2GetDeviceFromIndex (
            fc2Context context,
            unsigned int index,
            fc2PGRGuid * pGuid )
```

Gets the PGRGuid for a device.

It uniquely identifies the device specified by the index.

Parameters

context	The fc2Context to be used.
index	Zero based index of device.
pGuid	Unique PGRGuid for the device.

See also

fc2GetNumOfDevices()

Returns

An Error indicating the success or failure of the function.

6.1.2.11 fc2GetInterfaceTypeFromGuid()

```
FLYCAPTURE2_C_API fc2Error fc2GetInterfaceTypeFromGuid (
            fc2Context context,
            fc2PGRGuid * pGuid,
            fc2InterfaceType * pInterfaceType )
```

Gets the interface type associated with a PGRGuid.

This is useful in situations where there is a need to enumerate all cameras for a particular interface.

Parameters

context	The fc2Context to be used.
pGuid	The PGRGuid to get the interface for.
pInterfaceType	The interface type of the PGRGuid.

Returns

6.1.2.12 fc2GetNumOfCameras()

```
FLYCAPTURE2_C_API fc2Error fc2GetNumOfCameras (
            fc2Context context,
            unsigned int * pNumCameras )
```

Gets the number of cameras attached to the PC.

Parameters

context	The fc2Context to be used.
pNumCameras	Number of cameras detected.

Returns

A fc2Error indicating the success or failure of the function.

6.1.2.13 fc2GetNumOfDevices()

```
FLYCAPTURE2_C_API fc2Error fc2GetNumOfDevices (
             fc2Context context,
             unsigned int * pNumDevices )
```

Gets the number of devices.

This may include hubs, host controllers and other hardware devices (including cameras).

Parameters

context	The fc2Context to be used.
pNumDevices	The number of devices found.

Returns

An Error indicating the success or failure of the function.

6.1.2.14 fc2GetTopology()

```
FLYCAPTURE2_C_API fc2Error fc2GetTopology (
             fc2Context context,
             fc2TopologyNodeContext * pTopologyNodeContext )
```

Gets the topology information for the PC.

Parameters

context	The fc2Context to be used.
pTopologyNodeContext	A Topology Node context that will contain the topology information

Returns

An Error indicating the success or failure of the function.

6.1.2.15 fc2GetUsbLinkInfo()

```
FLYCAPTURE2_C_API fc2Error fc2GetUsbLinkInfo (
             fc2Context context,
             fc2PGRGuid guid,
             unsigned int * pValue )
```

Read usb link info for the port that the specified device is connected to.

Parameters

context	The fc2Context to be used.
guid	Unique PGRGuid for the device.
pValue	Value read from the card register.

Returns

An Error indicating the success or failure of the function.

6.1.2.16 fc2GetUsbPortStatus()

```
FLYCAPTURE2_C_API fc2Error fc2GetUsbPortStatus (
             fc2Context context,
             fc2PGRGuid guid,
             unsigned int * pValue )
```

Read usb port status for the port that the specified device is connected to.

Parameters

context	The fc2Context to be used.
guid	Unique PGRGuid for the device.
pValue	Value read from the card register.

Returns

An Error indicating the success or failure of the function.

6.1.2.17 fc2IsCameraControlable()

```
FLYCAPTURE2_C_API fc2Error fc2IsCameraControlable (
             fc2Context context,
             fc2PGRGuid * pGuid,
             BOOL * pControlable )
```

Query whether a GigE camera is controllable.

Parameters

context	The fc2Context to be used.
pGuid	Unique PGRGuid for the camera.
pControllable	True indicates camera is controllable

Returns

A fc2Error indicating the success or failure of the function.

6.1.2.18 fc2ReadPhyRegister()

```
FLYCAPTURE2_C_API fc2Error fc2ReadPhyRegister (
             fc2Context context,
             fc2PGRGuid guid,
             unsigned int page,
             unsigned int port,
             unsigned int address,
             unsigned int * pValue )
```

Read a phy register on the specified device.

The full address to be read from is determined by the page, port and address.

Parameters

context	The fc2Context to be used.
guid	Unique PGRGuid for the device.
page	Page to read from.
port	Port to read from.
address	Address to read from.
pValue	Value read from the phy register.

Returns

An Error indicating the success or failure of the function.

6.1.2.19 fc2RegisterCallback()

```
FLYCAPTURE2_C_API fc2Error fc2RegisterCallback (
             fc2Context context,
             fc2BusEventCallback enumCallback,
             fc2BusCallbackType callbackType,
             void * pParameter,
             fc2CallbackHandle * pCallbackHandle )
```

Register a callback function that will be called when the specified callback event occurs.

Parameters

context	The fc2Context to be used.
enumCallback	Pointer to function that will receive the callback.
callbackType	Type of callback to register for.
pParameter	Callback parameter to be passed to callback.
pCallbackHandle	Unique callback handle used for unregistering callback.

Returns

A fc2Error indicating the success or failure of the function.

6.1.2.20 fc2RescanBus()

```
FLYCAPTURE2_C_API fc2Error fc2RescanBus (
             fc2Context context )
```

Force a rescan of the buses.

This does not trigger a bus reset. The camera objects will be invalidated only if the camera network topology is changed (ie. a camera is disconnected or added)

Returns

An Error indicating the success or failure of the function.

6.1.2.21 fc2UnregisterCallback()

```
FLYCAPTURE2_C_API fc2Error fc2UnregisterCallback (
             fc2Context context,
             fc2CallbackHandle callbackHandle )
```

Unregister a callback function.

Parameters

context	The fc2Context to be used.
callbackHandle	Unique callback handle.

Returns

A fc2Error indicating the success or failure of the function.

6.1.2.22 fc2WritePhyRegister()

```
FLYCAPTURE2_C_API fc2Error fc2WritePhyRegister (
             fc2Context context,
             fc2PGRGuid guid,
             unsigned int page,
             unsigned int port,
             unsigned int address,
             unsigned int value )
```

Write a phy register on the specified device.

The full address to be written to is determined by the page, port and address.

Parameters

context	The fc2Context to be used.
guid	Unique PGRGuid for the device.
page	Page to write to.
port	Port to write to.
address	Address to write to.
value	Value to write to phy register.

Returns

An Error indicating the success or failure of the function.

6.2 Connection and Image Retrieval

These functions deal with connections and image retrieval from the camera.

Functions

- FLYCAPTURE2_C_API fc2Error fc2Connect (fc2Context context, fc2PGRGuid ∗guid)

  Connects the fc2Context to the camera specified by the GUID.
- FLYCAPTURE2_C_API fc2Error fc2Disconnect (fc2Context context)

  Disconnects the fc2Context from the camera.
- FLYCAPTURE2_C_API BOOL fc2IsConnected (fc2Context context)

  Checks if the fc2Context is connected to a physical camera specified by a GUID.
- FLYCAPTURE2_C_API fc2Error fc2SetCallback (fc2Context context, fc2ImageEventCallback pCallbackFn, void ∗pCallbackData)

  Sets the callback data to be used on completion of image transfer.
- FLYCAPTURE2_C_API fc2Error fc2StartCapture (fc2Context context)

  Starts isochronous image capture.
- FLYCAPTURE2_C_API fc2Error fc2StartCaptureCallback (fc2Context context, fc2ImageEventCallback p←CallbackFn, void ∗pCallbackData)

  Starts isochronous image capture.
- FLYCAPTURE2_C_API fc2Error fc2StartSyncCapture (unsigned int numCameras, fc2Context ∗pContexts)

  Starts synchronized isochronous image capture on multiple cameras.
- FLYCAPTURE2_C_API fc2Error fc2StartSyncCaptureCallback (unsigned int numCameras, fc2Context ∗p←Contexts, fc2ImageEventCallback ∗pCallbackFns, void ∗∗pCallbackDataArray)

  Starts synchronized isochronous image capture on multiple cameras.
- FLYCAPTURE2_C_API fc2Error fc2RetrieveBuffer (fc2Context context, fc2Image ∗pImage)

  Retrieves the next image object containing the next image.
- FLYCAPTURE2_C_API fc2Error fc2StopCapture (fc2Context context)

  Stops isochronous image transfer and cleans up all associated resources.
- FLYCAPTURE2_C_API fc2Error fc2WaitForBufferEvent (fc2Context context, fc2Image ∗pImage, unsigned int eventNumber)

  Retrieves the next image event containing the next part of the image.
- FLYCAPTURE2_C_API fc2Error fc2SetUserBuffers (fc2Context context, unsigned char ∗const ppMem←Buffers, int size, int nNumBuffers)

  Specify user allocated buffers to use as image data buffers.
- FLYCAPTURE2_C_API fc2Error fc2GetConfiguration (fc2Context context, fc2Config ∗config)

  Get the configuration associated with the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetConfiguration (fc2Context context, fc2Config ∗config)

  Set the configuration associated with the camera.

6.2.1 Detailed Description

These functions deal with connections and image retrieval from the camera.

6.2.2 Function Documentation

6.2.2.1 fc2Connect()

```
FLYCAPTURE2_C_API fc2Error fc2Connect (
             fc2Context context,
             fc2PGRGuid * guid )
```

Connects the fc2Context to the camera specified by the GUID.

Be aware that calling fc2CreateGUIContext() releases the CCP acquired for GigE cameras in fc2Connect(). Consider calling fc2Connect() after fc2CreateGUIContext().

Parameters

context	The fc2Context to be used.
guid	The unique identifier for a specific camera on the PC.

Returns

A fc2Error indicating the success or failure of the function.

6.2.2.2 fc2Disconnect()

```
FLYCAPTURE2_C_API fc2Error fc2Disconnect (
             fc2Context context )
```

Disconnects the fc2Context from the camera.

This allows another physical camera specified by a GUID to be connected to the fc2Context.

See also

fc2Connect()

Parameters

context	The fc2Context to be used.

Returns

A fc2Error indicating the success or failure of the function.

6.2.2.3 fc2GetConfiguration()

```
FLYCAPTURE2_C_API fc2Error fc2GetConfiguration (
             fc2Context context,
             fc2Config * config )
```

Get the configuration associated with the camera.

See also

fc2SetConfiguration()

Parameters

context	The fc2Context to be used.
config	Pointer to the configuration structure to be filled.

Returns

A fc2Error indicating the success or failure of the function.

6.2.2.4 fc2IsConnected()

```
FLYCAPTURE2_C_API BOOL fc2IsConnected (
            fc2Context context )
```

Checks if the fc2Context is connected to a physical camera specified by a GUID.

See also

fc2Connect()
fc2Disconnect()

Parameters

context	The fc2Context to be used.

Returns

Whether fc2Connect() was called on the fc2Context.

6.2.2.5 fc2RetrieveBuffer()

```
FLYCAPTURE2_C_API fc2Error fc2RetrieveBuffer (
            fc2Context context,
            fc2Image * pImage )
```

Retrieves the next image object containing the next image.

See also

fc2StartCapture()
fc2StopCapture()

Parameters

context	The fc2Context to be used.
pImage	Pointer to fc2Image to store image data.

Returns

A fc2Error indicating the success or failure of the function.

6.2.2.6 fc2SetCallback()

```
FLYCAPTURE2_C_API fc2Error fc2SetCallback (
              fc2Context context,
              fc2ImageEventCallback pCallbackFn,
              void * pCallbackData )
```

Sets the callback data to be used on completion of image transfer.

To clear the current stored callback data, pass in NULL for both callback arguments.

See also

fc2StartCapture()

Parameters

context	The fc2Context to be used.
pCallbackFn	A function to be called when a new image is received.
pCallbackData	A pointer to data that can be passed to the callback function.

Returns

A fc2Error indicating the success or failure of the function.

6.2.2.7 fc2SetConfiguration()

```
FLYCAPTURE2_C_API fc2Error fc2SetConfiguration (
              fc2Context context,
              fc2Config * config )
```

Set the configuration associated with the camera.

See also

fc2GetConfiguration()

Parameters

context	The fc2Context to be used.
config	Pointer to the configuration structure to be used.

Returns

A fc2Error indicating the success or failure of the function.

6.2.2.8 fc2SetUserBuffers()

```
FLYCAPTURE2_C_API fc2Error fc2SetUserBuffers (
            fc2Context context,
            unsigned char *const ppMemBuffers,
            int size,
            int nNumBuffers )
```

Specify user allocated buffers to use as image data buffers.

To prevent image tearing, the size of each buffer should be equal to ((unsigned int)(bufferSize + packetSize - 1)/packetSize) ∗ packetSize. The total size should be (size ∗ numBuffers) or larger. The packet Size that should be used differs between interfaces: Firewire: Use the Format7 packet size. Usb2: First round to Format7 packet size then round to 512 bytes. Usb3: Use a packet size of 1024 bytes. GigE: No need to do any rounding on GigE

See also

fc2StartCapture()
fc2RetrieveBuffer()
fc2StopCapture()

Parameters

context	The fc2Context to be used.
ppMemBuffers	Pointer to memory buffers to be written to. The size of the data should be equal to (size ∗ numBuffers) or larger.
size	The size of each buffer (in bytes).
nNumBuffers	Number of buffers in the array.

Returns

A fc2Error indicating the success or failure of the function.

6.2.2.9 fc2StartCapture()

```
FLYCAPTURE2_C_API fc2Error fc2StartCapture (
            fc2Context context )
```

Starts isochronous image capture.

It will use either the current video mode or the most recently set video mode of the camera.

See also

fc2RetrieveBuffer()
fc2StartSyncCapture()
fc2StopCapture()

Parameters

context	The fc2Context to be used.

Returns

A fc2Error indicating the success or failure of the function.

6.2.2.10 fc2StartCaptureCallback()

```
FLYCAPTURE2_C_API fc2Error fc2StartCaptureCallback (
             fc2Context context,
             fc2ImageEventCallback pCallbackFn,
             void * pCallbackData )
```

Starts isochronous image capture.

It will use either the current video mode or the most recently set video mode of the camera. The callback function is called when a new image is received from the camera.

See also

fc2RetrieveBuffer()
fc2StartSyncCapture()
fc2StopCapture()

Parameters

context	The fc2Context to be used.
pCallbackFn	A function to be called when a new image is received.
pCallbackData	A pointer to data that can be passed to the callback function. A NULL pointer is acceptable.

Returns

A fc2Error indicating the success or failure of the function.

6.2.2.11 fc2StartSyncCapture()

```
FLYCAPTURE2_C_API fc2Error fc2StartSyncCapture (
             unsigned int numCameras,
             fc2Context * pContexts )
```

Starts synchronized isochronous image capture on multiple cameras.

This function is only used for firewire cameras.

See also

fc2RetrieveBuffer()
fc2StartCapture()
fc2StopCapture()

Parameters

numCameras	Number of fc2Contexts in the ppCameras array.
pContexts	Array of fc2Contexts.

Returns

A fc2Error indicating the success or failure of the function.

6.2.2.12 fc2StartSyncCaptureCallback()

```
FLYCAPTURE2_C_API fc2Error fc2StartSyncCaptureCallback (
             unsigned int numCameras,
             fc2Context * pContexts,
             fc2ImageEventCallback * pCallbackFns,
             void ** pCallbackDataArray )
```

Starts synchronized isochronous image capture on multiple cameras.

This function is only used for firewire cameras.

See also

fc2RetrieveBuffer()
fc2StartCapture()
fc2StopCapture()

Parameters

numCameras	Number of fc2Contexts in the ppCameras array.
pContexts	Array of fc2Contexts.
pCallbackFns	Array of callback functions for each camera.
pCallbackDataArray	Array of callback data pointers.

Returns

A fc2Error indicating the success or failure of the function.

6.2.2.13 fc2StopCapture()

```
FLYCAPTURE2_C_API fc2Error fc2StopCapture (
            fc2Context context )
```

Stops isochronous image transfer and cleans up all associated resources.

See also

fc2StartCapture()
fc2RetrieveBuffer()

Parameters

context	The fc2Context to be used.

Returns

A fc2Error indicating the success or failure of the function.

6.2.2.14 fc2WaitForBufferEvent()

```
FLYCAPTURE2_C_API fc2Error fc2WaitForBufferEvent (
            fc2Context context,
            fc2Image * pImage,
            unsigned int eventNumber )
```

Retrieves the next image event containing the next part of the image.

See also

fc2StartCapture()
fc2RetrieveBuffer()
fc2StopCapture()

Parameters

context	The fc2Context to be used.
pImage	Pointer to fc2Image to store image data.
eventNumber	The event number to wait for.

Returns

An Error indicating the success or failure of the function.

6.3 Information and Properties

These functions deal with information and properties can be retrieved from the camera.

Functions

- FLYCAPTURE2_C_API fc2Error fc2GetCameraInfo (fc2Context context, fc2CameraInfo ∗pCameraInfo)

  Retrieves information from the camera such as serial number, model name and other camera information.
- FLYCAPTURE2_C_API fc2Error fc2GetPropertyInfo (fc2Context context, fc2PropertyInfo ∗propInfo)

  Retrieves information about the specified camera property.
- FLYCAPTURE2_C_API fc2Error fc2GetProperty (fc2Context context, fc2Property ∗prop)

  Reads the settings for the specified property from the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetProperty (fc2Context context, fc2Property ∗prop)

  Writes the settings for the specified property to the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetPropertyBroadcast (fc2Context context, fc2Property ∗prop)

  Writes the settings for the specified property to the camera.

6.3.1 Detailed Description

These functions deal with information and properties can be retrieved from the camera.

6.3.2 Function Documentation

6.3.2.1 fc2GetCameraInfo()

```
FLYCAPTURE2_C_API fc2Error fc2GetCameraInfo (
             fc2Context context,
             fc2CameraInfo * pCameraInfo )
```

Retrieves information from the camera such as serial number, model name and other camera information.

Parameters

context	The fc2Context to be used.
pCameraInfo	Pointer to the camera information structure to be filled.

Returns

A fc2Error indicating the success or failure of the function.

6.3.2.2 fc2GetProperty()

```
FLYCAPTURE2_C_API fc2Error fc2GetProperty (
            fc2Context context,
            fc2Property * prop )
```

Reads the settings for the specified property from the camera.

The property type must be specified in the fc2Property structure passed into the function in order for the function to succeed. If auto is on, the integer and abs values returned may not be consistent with each other.

See also

fc2GetPropertyInfo()
fc2SetProperty()

Parameters

context	The fc2Context to be used.
prop	Pointer to the Property structure to be filled.

Returns

A fc2Error indicating the success or failure of the function.

6.3.2.3 fc2GetPropertyInfo()

```
FLYCAPTURE2_C_API fc2Error fc2GetPropertyInfo (
            fc2Context context,
            fc2PropertyInfo * propInfo )
```

Retrieves information about the specified camera property.

The property type must be specified in the fc2PropertyInfo structure passed into the function in order for the function to succeed.

See also

fc2GetProperty()
fc2SetProperty()

Parameters

context	The fc2Context to be used.
propInfo	Pointer to the PropertyInfo structure to be filled.

Returns

A fc2Error indicating the success or failure of the function.

6.3.2.4 fc2SetProperty()

```
FLYCAPTURE2_C_API fc2Error fc2SetProperty (
            fc2Context context,
            fc2Property * prop )
```

Writes the settings for the specified property to the camera.

The property type must be specified in the Property structure passed into the function in order for the function to succeed. The absControl flag controls whether the absolute or integer value is written to the camera. Use fc2GetPropertyInfo() to query which options are available for a specific property.

See also

fc2GetPropertyInfo()
fc2GetProperty()

Parameters

context	The fc2Context to be used.
prop	Pointer to the Property structure to be used.

Returns

A fc2Error indicating the success or failure of the function.

6.3.2.5 fc2SetPropertyBroadcast()

```
FLYCAPTURE2_C_API fc2Error fc2SetPropertyBroadcast (
            fc2Context context,
            fc2Property * prop )
```

Writes the settings for the specified property to the camera.

The property type must be specified in the Property structure passed into the function in order for the function to succeed. The absControl flag controls whether the absolute or integer value is written to the camera.

Parameters

context	The fc2Context to be used.
prop	Pointer to the Property structure to be used.

Returns

A fc2Error indicating the success or failure of the function.

6.4 General Purpose Input / Output

These functions deal with general GPIO pin control on the camera.

Functions

- FLYCAPTURE2_C_API fc2Error fc2GetGPIOPinDirection (fc2Context context, unsigned int pin, unsigned int ∗pDirection)

  Get the GPIO pin direction for the specified pin.
- FLYCAPTURE2_C_API fc2Error fc2SetGPIOPinDirection (fc2Context context, unsigned int pin, unsigned int direction)

  Set the GPIO pin direction for the specified pin.
- FLYCAPTURE2_C_API fc2Error fc2SetGPIOPinDirectionBroadcast (fc2Context context, unsigned int pin, unsigned int direction)

  Set the GPIO pin direction for the specified pin.

6.4.1 Detailed Description

These functions deal with general GPIO pin control on the camera.

6.4.2 Function Documentation

6.4.2.1 fc2GetGPIOPinDirection()

```
FLYCAPTURE2_C_API fc2Error fc2GetGPIOPinDirection (
            fc2Context context,
            unsigned int pin,
            unsigned int * pDirection )
```

Get the GPIO pin direction for the specified pin.

This is not a required call when using the trigger or strobe functions as the pin direction is set automatically internally.

See also

fc2SetGPIOPinDirection()
fc2SetGPIOPinDirectionBroadcast()

Parameters

context	The fc2Context to be used.
pin	Pin to get the direction for.
pDirection	Direction of the pin. 0 for input, 1 for output.

Returns

A fc2Error indicating the success or failure of the function.

6.4.2.2 fc2SetGPIOPinDirection()

```
FLYCAPTURE2_C_API fc2Error fc2SetGPIOPinDirection (
            fc2Context context,
            unsigned int pin,
            unsigned int direction )
```

Set the GPIO pin direction for the specified pin.

This is useful if there is a need to set the pin into an input pin (i.e. to read the voltage) off the pin without setting it as a trigger source. This is not a required call when using the trigger or strobe functions as the pin direction is set automatically internally.

See also

fc2GetGPIOPinDirection()
fc2SetGPIOPinDirectionBroadcast()

Parameters

context	The fc2Context to be used.
pin	Pin to get the direction for.
direction	Direction of the pin. 0 for input, 1 for output.

Returns

A fc2Error indicating the success or failure of the function.

6.4.2.3 fc2SetGPIOPinDirectionBroadcast()

```
FLYCAPTURE2_C_API fc2Error fc2SetGPIOPinDirectionBroadcast (
            fc2Context context,
            unsigned int pin,
            unsigned int direction )
```

Set the GPIO pin direction for the specified pin.

This is useful if there is a need to set the pin into an input pin (i.e. to read the voltage) off the pin without setting it as a trigger source. This is not a required call when using the trigger or strobe functions as the pin direction is set automatically internally.

See also

fc2GetGPIOPinDirection()

Parameters

context	The fc2Context to be used.
pin	Pin to get the direction for.
direction	Direction of the pin. 0 for input, 1 for output.

Returns

A fc2Error indicating the success or failure of the function.

6.5 Trigger

These functions deal with trigger control on the camera.

Functions

- FLYCAPTURE2_C_API fc2Error fc2GetTriggerModeInfo (fc2Context context, fc2TriggerModeInfo ∗trigger↩ ModeInfo)

    Retrieve trigger information from the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetTriggerMode (fc2Context context, fc2TriggerMode ∗triggerMode)

    Retrieve current trigger settings from the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetTriggerMode (fc2Context context, fc2TriggerMode ∗triggerMode)

    Set the specified trigger settings to the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetTriggerModeBroadcast (fc2Context context, fc2TriggerMode ∗triggerMode)

    Set the specified trigger settings to the camera.
- FLYCAPTURE2_C_API fc2Error fc2FireSoftwareTrigger (fc2Context context)

    Fire the software trigger according to the DCAM specifications.
- FLYCAPTURE2_C_API fc2Error fc2FireSoftwareTriggerBroadcast (fc2Context context)

    Fire the software trigger according to the DCAM specifications.
- FLYCAPTURE2_C_API fc2Error fc2GetTriggerDelayInfo (fc2Context context, fc2TriggerDelayInfo ∗trigger↩ DelayInfo)

    Retrieve trigger delay information from the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetTriggerDelay (fc2Context context, fc2TriggerDelay ∗triggerDelay)

    Retrieve current trigger delay settings from the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetTriggerDelay (fc2Context context, fc2TriggerDelay ∗triggerDelay)

    Set the specified trigger delay settings to the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetTriggerDelayBroadcast (fc2Context context, fc2TriggerDelay ∗triggerDelay)

    Set the specified trigger delay settings to the camera.

6.5.1 Detailed Description

These functions deal with trigger control on the camera.

6.5.2 Function Documentation

6.5.2.1 fc2FireSoftwareTrigger()

```
FLYCAPTURE2_C_API fc2Error fc2FireSoftwareTrigger (
            fc2Context context )
```

Fire the software trigger according to the DCAM specifications.

Parameters

context	The fc2Context to be used.

Returns

A fc2Error indicating the success or failure of the function.

6.5.2.2 fc2FireSoftwareTriggerBroadcast()

```
FLYCAPTURE2_C_API fc2Error fc2FireSoftwareTriggerBroadcast (
              fc2Context context )
```

Fire the software trigger according to the DCAM specifications.

Parameters

context	The fc2Context to be used.

Returns

A fc2Error indicating the success or failure of the function.

6.5.2.3 fc2GetTriggerDelay()

```
FLYCAPTURE2_C_API fc2Error fc2GetTriggerDelay (
              fc2Context context,
              fc2TriggerDelay * triggerDelay )
```

Retrieve current trigger delay settings from the camera.

See also

fc2GetTriggerModeInfo()
fc2GetTriggerMode()
fc2SetTriggerMode()
fc2GetTriggerDelayInfo()
fc2SetTriggerDelay()
fc2SetTriggerDelayBroadcast()

Parameters

context	The fc2Context to be used.
triggerDelay	Structure to receive trigger delay settings.

Returns

A fc2Error indicating the success or failure of the function.

6.5.2.4 fc2GetTriggerDelayInfo()

```
FLYCAPTURE2_C_API fc2Error fc2GetTriggerDelayInfo (
            fc2Context context,
            fc2TriggerDelayInfo * triggerDelayInfo )
```

Retrieve trigger delay information from the camera.

See also

fc2GetTriggerModeInfo()
fc2GetTriggerMode()
fc2SetTriggerMode()
fc2GetTriggerDelay()
fc2SetTriggerDelay()
fc2SetTriggerDelayBroadcast()

Parameters

context	The fc2Context to be used.
triggerDelayInfo	Structure to receive trigger delay information.

Returns

A fc2Error indicating the success or failure of the function.

6.5.2.5 fc2GetTriggerMode()

```
FLYCAPTURE2_C_API fc2Error fc2GetTriggerMode (
            fc2Context context,
            fc2TriggerMode * triggerMode )
```

Retrieve current trigger settings from the camera.

See also

fc2GetTriggerModeInfo()
fc2SetTriggerMode()
fc2SetTriggerModeBroadcast()
fc2GetTriggerDelayInfo()
fc2GetTriggerDelay()
fc2SetTriggerDelay()

Parameters

context	The fc2Context to be used.
triggerMode	Structure to receive trigger mode settings.

Returns

A fc2Error indicating the success or failure of the function.

6.5.2.6 fc2GetTriggerModeInfo()

```
FLYCAPTURE2_C_API fc2Error fc2GetTriggerModeInfo (
            fc2Context context,
            fc2TriggerModeInfo * triggerModeInfo )
```

Retrieve trigger information from the camera.

See also

fc2GetTriggerMode()
fc2SetTriggerMode()
fc2SetTriggerModeBroadcast()
fc2GetTriggerDelayInfo()
fc2GetTriggerDelay()
fc2SetTriggerDelay()

Parameters

context	The fc2Context to be used.
triggerModeInfo	Structure to receive trigger information.

Returns

A fc2Error indicating the success or failure of the function.

6.5.2.7 fc2SetTriggerDelay()

```
FLYCAPTURE2_C_API fc2Error fc2SetTriggerDelay (
            fc2Context context,
            fc2TriggerDelay * triggerDelay )
```

Set the specified trigger delay settings to the camera.

See also

fc2GetTriggerModeInfo()
fc2GetTriggerMode()
fc2SetTriggerMode()
fc2GetTriggerDelayInfo()
fc2GetTriggerDelay()
fc2SetTriggerDelayBroadcast()

Parameters

context	The fc2Context to be used.
triggerDelay	Structure providing trigger delay settings.

Returns

A fc2Error indicating the success or failure of the function.

6.5.2.8 fc2SetTriggerDelayBroadcast()

```
FLYCAPTURE2_C_API fc2Error fc2SetTriggerDelayBroadcast (
            fc2Context context,
            fc2TriggerDelay * triggerDelay )
```

Set the specified trigger delay settings to the camera.

See also

fc2GetTriggerModeInfo()
fc2GetTriggerMode()
fc2SetTriggerMode()
fc2GetTriggerDelayInfo()
fc2GetTriggerDelay()
fc2SetTriggerDelay()

Parameters

context	The fc2Context to be used.
triggerDelay	Structure providing trigger delay settings.

Returns

A fc2Error indicating the success or failure of the function.

6.5.2.9 fc2SetTriggerMode()

```
FLYCAPTURE2_C_API fc2Error fc2SetTriggerMode (
            fc2Context context,
            fc2TriggerMode * triggerMode )
```

Set the specified trigger settings to the camera.

See also

fc2GetTriggerModeInfo()
fc2GetTriggerMode()
fc2GetTriggerDelayInfo()
fc2GetTriggerDelay()
fc2SetTriggerDelay()
fc2SetTriggerModeBroadcast()

Parameters

context	The fc2Context to be used.
triggerMode	Structure providing trigger mode settings.

Returns

A fc2Error indicating the success or failure of the function.

6.5.2.10 fc2SetTriggerModeBroadcast()

```
FLYCAPTURE2_C_API fc2Error fc2SetTriggerModeBroadcast (
            fc2Context context,
            fc2TriggerMode * triggerMode )
```

Set the specified trigger settings to the camera.

See also

fc2GetTriggerModeInfo()
fc2GetTriggerMode()
fc2GetTriggerDelayInfo()
fc2GetTriggerDelay()
fc2SetTriggerDelay()
fc2SetTriggerMode()

Parameters

context	The fc2Context to be used.
triggerMode	Structure providing trigger mode settings.

Returns

A fc2Error indicating the success or failure of the function.

6.6 Strobe

These functions deal with strobe control on the camera.

Functions

- FLYCAPTURE2_C_API fc2Error fc2GetStrobeInfo (fc2Context context, fc2StrobeInfo ∗strobeInfo)

  Retrieve strobe information from the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetStrobe (fc2Context context, fc2StrobeControl ∗strobeControl)

  Retrieve current strobe settings from the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetStrobe (fc2Context context, fc2StrobeControl ∗strobeControl)

  Set current strobe settings to the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetStrobeBroadcast (fc2Context context, fc2StrobeControl ∗strobe←Control)

  Set current strobe settings to the camera.

6.6.1 Detailed Description

These functions deal with strobe control on the camera.

6.6.2 Function Documentation

6.6.2.1 fc2GetStrobe()

```
FLYCAPTURE2_C_API fc2Error fc2GetStrobe (
            fc2Context context,
            fc2StrobeControl * strobeControl )
```

Retrieve current strobe settings from the camera.

The strobe pin must be specified in the structure before being passed in to the function.

See also

fc2GetStrobeInfo()
fc2SetStrobe()
fc2SetStrobeBroadcast()

Parameters

context	The fc2Context to be used.
strobeControl	Structure to receive strobe settings.

Returns

A fc2Error indicating the success or failure of the function.

6.6.2.2 fc2GetStrobeInfo()

```
FLYCAPTURE2_C_API fc2Error fc2GetStrobeInfo (
            fc2Context context,
            fc2StrobeInfo * strobeInfo )
```

Retrieve strobe information from the camera.

See also

fc2GetStrobe()
fc2SetStrobe()
fc2SetStrobeBroadcast()

Parameters

context	The fc2Context to be used.
strobeInfo	Structure to receive strobe information.

Returns

A fc2Error indicating the success or failure of the function.

6.6.2.3 fc2SetStrobe()

```
FLYCAPTURE2_C_API fc2Error fc2SetStrobe (
            fc2Context context,
            fc2StrobeControl * strobeControl )
```

Set current strobe settings to the camera.

The strobe pin must be specified in the structure before being passed in to the function.

See also

fc2GetStrobeInfo()
fc2GetStrobe()
fc2SetStrobeBroadcast()

Parameters

context	The fc2Context to be used.
strobeControl	Structure providing strobe settings.

Returns

A fc2Error indicating the success or failure of the function.

6.6.2.4 fc2SetStrobeBroadcast()

```
FLYCAPTURE2_C_API fc2Error fc2SetStrobeBroadcast (
            fc2Context context,
            fc2StrobeControl * strobeControl )
```

Set current strobe settings to the camera.

The strobe pin must be specified in the structure before being passed in to the function.

See also

fc2GetStrobeInfo()
fc2GetStrobe()
fc2SetStrobe()

Parameters

context	The fc2Context to be used.
strobeControl	Structure providing strobe settings.

Returns

A fc2Error indicating the success or failure of the function.

6.7 Look Up Table

These functions deal with Look Up Table control on the camera.

Functions

- FLYCAPTURE2_C_API fc2Error fc2GetLUTInfo (fc2Context context, fc2LUTData ∗pData)

  Query if LUT support is available on the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetLUTBankInfo (fc2Context context, unsigned int bank, BOOL ∗p←ReadSupported, BOOL ∗pWriteSupported)

  Query the read/write status of a single LUT bank.
- FLYCAPTURE2_C_API fc2Error fc2GetActiveLUTBank (fc2Context context, unsigned int ∗pActiveBank)

  Get the LUT bank that is currently being used.
- FLYCAPTURE2_C_API fc2Error fc2SetActiveLUTBank (fc2Context context, unsigned int activeBank)

  Set the LUT bank that will be used.
- FLYCAPTURE2_C_API fc2Error fc2EnableLUT (fc2Context context, BOOL on)

  Enable or disable LUT functionality on the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetLUTChannel (fc2Context context, unsigned int bank, unsigned int channel, unsigned int sizeEntries, unsigned int ∗pEntries)

  Get the LUT channel settings from the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetLUTChannel (fc2Context context, unsigned int bank, unsigned int channel, unsigned int sizeEntries, unsigned int ∗pEntries)

  Set the LUT channel settings to the camera.

6.7.1 Detailed Description

These functions deal with Look Up Table control on the camera.

6.7.2 Function Documentation

6.7.2.1 fc2EnableLUT()

```
FLYCAPTURE2_C_API fc2Error fc2EnableLUT (
             fc2Context context,
             BOOL on )
```

Enable or disable LUT functionality on the camera.

See also

fc2GetLUTInfo()
fc2GetLUTChannel()
fc2SetLUTChannel()

Parameters

context	The fc2Context to be used.
on	Whether to enable or disable LUT.

Returns

A fc2Error indicating the success or failure of the function.

6.7.2.2 fc2GetActiveLUTBank()

```
FLYCAPTURE2_C_API fc2Error fc2GetActiveLUTBank (
             fc2Context context,
             unsigned int * pActiveBank )
```

Get the LUT bank that is currently being used.

For cameras with PGR LUT, the active bank is always 0.

Parameters

context	The fc2Context to be used.
pActiveBank	The currently active bank.

Returns

A fc2Error indicating the success or failure of the function.

6.7.2.3 fc2GetLUTBankInfo()

```
FLYCAPTURE2_C_API fc2Error fc2GetLUTBankInfo (
             fc2Context context,
             unsigned int bank,
             BOOL * pReadSupported,
             BOOL * pWriteSupported )
```

Query the read/write status of a single LUT bank.

Parameters

context	The fc2Context to be used.
bank	The bank to query.
pReadSupported	Whether reading from the bank is supported.
pWriteSupported	Whether writing to the bank is supported.

Returns

A fc2Error indicating the success or failure of the function.

6.7.2.4 fc2GetLUTChannel()

```
FLYCAPTURE2_C_API fc2Error fc2GetLUTChannel (
            fc2Context context,
            unsigned int bank,
            unsigned int channel,
            unsigned int sizeEntries,
            unsigned int * pEntries )
```

Get the LUT channel settings from the camera.

See also

fc2GetLUTInfo()
fc2EnableLUT()
fc2SetLUTChannel()

Parameters

context	The fc2Context to be used.
bank	Bank to retrieve.
channel	Channel to retrieve.
sizeEntries	Number of entries in LUT table to read.
pEntries	Array to store LUT entries.

Returns

A fc2Error indicating the success or failure of the function.

6.7.2.5 fc2GetLUTInfo()

```
FLYCAPTURE2_C_API fc2Error fc2GetLUTInfo (
            fc2Context context,
            fc2LUTData * pData )
```

Query if LUT support is available on the camera.

Note that some cameras may report support for the LUT and return an inputBitDepth of 0. In these cases use log2(numEntries) for the inputBitDepth.

See also

fc2EnableLUT()
fc2GetLUTChannel()
fc2SetLUTChannel()

Parameters

context	The fc2Context to be used.
pData	The LUT structure to be filled.

Returns

A fc2Error indicating the success or failure of the function.

6.7.2.6 fc2SetActiveLUTBank()

```
FLYCAPTURE2_C_API fc2Error fc2SetActiveLUTBank (
            fc2Context context,
            unsigned int activeBank )
```

Set the LUT bank that will be used.

Parameters

context	The fc2Context to be used.
activeBank	The bank to be set as active.

Returns

A fc2Error indicating the success or failure of the function.

6.7.2.7 fc2SetLUTChannel()

```
FLYCAPTURE2_C_API fc2Error fc2SetLUTChannel (
            fc2Context context,
            unsigned int bank,
            unsigned int channel,
            unsigned int sizeEntries,
            unsigned int * pEntries )
```

Set the LUT channel settings to the camera.

See also

fc2GetLUTInfo()
fc2EnableLUT()
fc2GetLUTChannel()

Parameters

context	The fc2Context to be used.
bank	Bank to set.
channel	Channel to set.
sizeEntries	Number of entries in LUT table to write. This must be the same size as numEntries returned by GetLutInfo().
pEntries	Array containing LUT entries to write.

Returns

A fc2Error indicating the success or failure of the function.

6.8 Memory Channels

These functions deal with memory channel control on the camera.

Functions

- FLYCAPTURE2_C_API fc2Error fc2GetMemoryChannel (fc2Context context, unsigned int *pCurrent↩ Channel)

  Retrieve the current memory channel from the camera.
- FLYCAPTURE2_C_API fc2Error fc2SaveToMemoryChannel (fc2Context context, unsigned int channel)

  Save the current settings to the specified current memory channel.
- FLYCAPTURE2_C_API fc2Error fc2RestoreFromMemoryChannel (fc2Context context, unsigned int channel)

  Restore the specified current memory channel.
- FLYCAPTURE2_C_API fc2Error fc2GetMemoryChannelInfo (fc2Context context, unsigned int *pNum↩ Channels)

  Query the camera for memory channel support.
- FLYCAPTURE2_C_API fc2Error fc2GetEmbeddedImageInfo (fc2Context context, fc2EmbeddedImageInfo *pInfo)

  Get the current status of the embedded image information register, as well as the availability of each embedded property.
- FLYCAPTURE2_C_API fc2Error fc2SetEmbeddedImageInfo (fc2Context context, fc2EmbeddedImageInfo *pInfo)

  Sets the on/off values of the embedded image information structure to the camera.

6.8.1 Detailed Description

These functions deal with memory channel control on the camera.

6.8.2 Function Documentation

6.8.2.1 fc2GetEmbeddedImageInfo()

```
FLYCAPTURE2_C_API fc2Error fc2GetEmbeddedImageInfo (
            fc2Context context,
            fc2EmbeddedImageInfo * pInfo )
```

Get the current status of the embedded image information register, as well as the availability of each embedded property.

See also

fc2SetEmbeddedImageInfo()

Parameters

context	The fc2Context to be used.
pInfo	Structure to be filled.

Returns

A fc2Error indicating the success or failure of the function.

6.8.2.2 fc2GetMemoryChannel()

```
FLYCAPTURE2_C_API fc2Error fc2GetMemoryChannel (
            fc2Context context,
            unsigned int * pCurrentChannel )
```

Retrieve the current memory channel from the camera.

See also

fc2SaveToMemoryChannel()
fc2RestoreFromMemoryChannel()
fc2GetMemoryChannelInfo()

Parameters

context	The fc2Context to be used.
pCurrentChannel	Current memory channel.

Returns

A fc2Error indicating the success or failure of the function.

6.8.2.3 fc2GetMemoryChannelInfo()

```
FLYCAPTURE2_C_API fc2Error fc2GetMemoryChannelInfo (
            fc2Context context,
            unsigned int * pNumChannels )
```

Query the camera for memory channel support.

If the number of channels are 0, then memory channel support is not available.

See also

fc2GetMemoryChannel()
fc2SaveToMemoryChannel()
fc2RestoreFromMemoryChannel()

Parameters

context	The fc2Context to be used.
pNumChannels	Number of memory channels supported.

Returns

A fc2Error indicating the success or failure of the function.

6.8.2.4 fc2RestoreFromMemoryChannel()

```
FLYCAPTURE2_C_API fc2Error fc2RestoreFromMemoryChannel (
            fc2Context context,
            unsigned int channel )
```

Restore the specified current memory channel.

See also

fc2GetMemoryChannel()
fc2SaveToMemoryChannel()
fc2GetMemoryChannelInfo()

Parameters

context	The fc2Context to be used.
channel	Memory channel to restore from.

Returns

A fc2Error indicating the success or failure of the function.

6.8.2.5 fc2SaveToMemoryChannel()

```
FLYCAPTURE2_C_API fc2Error fc2SaveToMemoryChannel (
            fc2Context context,
            unsigned int channel )
```

Save the current settings to the specified current memory channel.

See also

fc2GetMemoryChannel()
fc2RestoreFromMemoryChannel()
fc2GetMemoryChannelInfo()

Parameters

context	The fc2Context to be used.
channel	Memory channel to save to.

Returns

A fc2Error indicating the success or failure of the function.

6.8.2.6 fc2SetEmbeddedImageInfo()

```
FLYCAPTURE2_C_API fc2Error fc2SetEmbeddedImageInfo (
            fc2Context context,
            fc2EmbeddedImageInfo * pInfo )
```

Sets the on/off values of the embedded image information structure to the camera.

See also

fc2GetEmbeddedImageInfo()

Parameters

context	The fc2Context to be used.
pInfo	Structure to be used.

Returns

A fc2Error indicating the success or failure of the function.

6.9 Register Operation

These functions deal with register operation on the camera.

Functions

- FLYCAPTURE2_C_API fc2Error fc2WriteRegister (fc2Context context, unsigned int address, unsigned int value)

  Write to the specified register on the camera.
- FLYCAPTURE2_C_API fc2Error fc2ReadRegister (fc2Context context, unsigned int address, unsigned int ∗pValue)

  Read the specified register from the camera.
- FLYCAPTURE2_C_API fc2Error fc2WriteRegisterBroadcast (fc2Context context, unsigned int address, unsigned int value)

  Write to the specified register on the camera with broadcast.
- FLYCAPTURE2_C_API fc2Error fc2WriteRegisterBlock (fc2Context context, unsigned short addressHigh, unsigned int addressLow, const unsigned int ∗pBuffer, unsigned int length)

  Write to the specified register block on the camera.
- FLYCAPTURE2_C_API fc2Error fc2ReadRegisterBlock (fc2Context context, unsigned short addressHigh, unsigned int addressLow, unsigned int ∗pBuffer, unsigned int length)

  Write to the specified register block on the camera.
- FLYCAPTURE2_C_API const char ∗ fc2GetRegisterString (unsigned int registerVal)

  Returns a text representation of the register value.

6.9.1 Detailed Description

These functions deal with register operation on the camera.

6.9.2 Function Documentation

6.9.2.1 fc2GetRegisterString()

```
FLYCAPTURE2_C_API const char* fc2GetRegisterString (
            unsigned int registerVal )
```

Returns a text representation of the register value.

Parameters

registerVal	The register value to query.

Returns

A fc2Error indicating the success or failure of the function.

6.9.2.2 fc2ReadRegister()

```
FLYCAPTURE2_C_API fc2Error fc2ReadRegister (
            fc2Context context,
            unsigned int address,
            unsigned int * pValue )
```

Read the specified register from the camera.

See also

fc2WriteRegister()

Parameters

context	The fc2Context to be used.
address	DCAM address to be read from.
pValue	The value that is read.

Returns

A fc2Error indicating the success or failure of the function.

6.9.2.3 fc2ReadRegisterBlock()

```
FLYCAPTURE2_C_API fc2Error fc2ReadRegisterBlock (
            fc2Context context,
            unsigned short addressHigh,
            unsigned int addressLow,
            unsigned int * pBuffer,
            unsigned int length )
```

Write to the specified register block on the camera.

See also

fc2WriteRegisterBlock()

Parameters

context	The fc2Context to be used.
addressHigh	Top 16 bits of the 48-bit absolute address to read from.
addressLow	Bottom 32 bits of the 48 bits absolute address to read from.
pBuffer	Array to store read data.
length	Size of array, in quadlets.

Returns

A fc2Error indicating the success or failure of the function.

6.9.2.4 fc2WriteRegister()

```
FLYCAPTURE2_C_API fc2Error fc2WriteRegister (
            fc2Context context,
            unsigned int address,
            unsigned int value )
```

Write to the specified register on the camera.

See also

fc2ReadRegister()

Parameters

context	The fc2Context to be used.
address	DCAM address to be written to.
value	The value to be written.

Returns

A fc2Error indicating the success or failure of the function.

6.9.2.5 fc2WriteRegisterBlock()

```
FLYCAPTURE2_C_API fc2Error fc2WriteRegisterBlock (
            fc2Context context,
            unsigned short addressHigh,
            unsigned int addressLow,
            const unsigned int * pBuffer,
            unsigned int length )
```

Write to the specified register block on the camera.

See also

fc2ReadRegisterBlock()

Parameters

context	The fc2Context to be used.
addressHigh	Top 16 bits of the 48-bit absolute address to write to.
addressLow	Bottom 32 bits of the 48 bits absolute address to write to.
pBuffer	Array containing data to be written.
length	Size of array, in quadlets.

Returns

A fc2Error indicating the success or failure of the function.

6.9.2.6 fc2WriteRegisterBroadcast()

```
FLYCAPTURE2_C_API fc2Error fc2WriteRegisterBroadcast (
            fc2Context context,
            unsigned int address,
            unsigned int value )
```

Write to the specified register on the camera with broadcast.

See also

fc2ReadRegisterBlock()

Parameters

context	The fc2Context to be used.
address	DCAM address to be written to.
value	The value to be written.

Returns

A fc2Error indicating the success or failure of the function.

6.10 DCAM Formats

These functions deal with DCAM video mode and frame rate on the camera.

Functions

- FLYCAPTURE2_C_API fc2Error fc2GetVideoModeAndFrameRateInfo (fc2Context context, fc2VideoMode videoMode, fc2FrameRate frameRate, BOOL ∗pSupported)

  Query the camera to determine if the specified video mode and frame rate is supported.
- FLYCAPTURE2_C_API fc2Error fc2GetVideoModeAndFrameRate (fc2Context context, fc2VideoMode ∗videoMode, fc2FrameRate ∗frameRate)

  Get the current video mode and frame rate from the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetVideoModeAndFrameRate (fc2Context context, fc2VideoMode videoMode, fc2FrameRate frameRate)

  Set the specified video mode and frame rate to the camera.

6.10.1 Detailed Description

These functions deal with DCAM video mode and frame rate on the camera.

This is only used for firewire and usb2 cameras.

6.10.2 Function Documentation

6.10.2.1 fc2GetVideoModeAndFrameRate()

```
FLYCAPTURE2_C_API fc2Error fc2GetVideoModeAndFrameRate (
            fc2Context context,
            fc2VideoMode ∗ videoMode,
            fc2FrameRate ∗ frameRate )
```

Get the current video mode and frame rate from the camera.

If the camera is in Format7, the video mode will be VIDEOMODE_FORMAT7 and the frame rate will be FRAME↩ RATE_FORMAT7.

Parameters

context	The fc2Context to be used.
videoMode	Current video mode.
frameRate	Current frame rate.

Returns

A fc2Error indicating the success or failure of the function.

6.10.2.2 fc2GetVideoModeAndFrameRateInfo()

```
FLYCAPTURE2_C_API fc2Error fc2GetVideoModeAndFrameRateInfo (
             fc2Context context,
             fc2VideoMode videoMode,
             fc2FrameRate frameRate,
             BOOL * pSupported )
```

Query the camera to determine if the specified video mode and frame rate is supported.

Parameters

context	The fc2Context to be used.
videoMode	Video mode to check.
frameRate	Frame rate to check.
pSupported	Whether the video mode and frame rate is supported.

Returns

A fc2Error indicating the success or failure of the function.

6.10.2.3 fc2SetVideoModeAndFrameRate()

```
FLYCAPTURE2_C_API fc2Error fc2SetVideoModeAndFrameRate (
             fc2Context context,
             fc2VideoMode videoMode,
             fc2FrameRate frameRate )
```

Set the specified video mode and frame rate to the camera.

It is not possible to set the camera to VIDEOMODE_FORMAT7 or FRAMERATE_FORMAT7. Use the Format7 functions to set the camera into Format7.

Parameters

context	The fc2Context to be used.
videoMode	Video mode to set to camera.
frameRate	Frame rate to set to camera.

Returns

A fc2Error indicating the success or failure of the function.

6.11 Format7

These functions deal with Format7 custom image control on the camera.

Functions

- FLYCAPTURE2_C_API fc2Error fc2GetFormat7Info (fc2Context context, fc2Format7Info ∗info, BOOL ∗p↩ Supported)

  Retrieve the availability of Format7 custom image mode and the camera capabilities for the specified Format7 mode.
- FLYCAPTURE2_C_API fc2Error fc2ValidateFormat7Settings (fc2Context context, fc2Format7ImageSettings ∗imageSettings, BOOL ∗settingsAreValid, fc2Format7PacketInfo ∗packetInfo)

  Validates Format7ImageSettings structure and returns valid packet size information if the image settings are valid.
- FLYCAPTURE2_C_API fc2Error fc2GetFormat7Configuration (fc2Context context, fc2Format7Image↩ Settings ∗imageSettings, unsigned int ∗packetSize, float ∗percentage)

  Get the current Format7 configuration from the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetFormat7ConfigurationPacket (fc2Context context, fc2Format7↩ ImageSettings ∗imageSettings, unsigned int packetSize)

  Set the current Format7 configuration to the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetFormat7Configuration (fc2Context context, fc2Format7ImageSettings ∗imageSettings, float percentSpeed)

  Set the current Format7 configuration to the camera.

6.11.1 Detailed Description

These functions deal with Format7 custom image control on the camera.

6.11.2 Function Documentation

6.11.2.1 fc2GetFormat7Configuration()

```
FLYCAPTURE2_C_API fc2Error fc2GetFormat7Configuration (
                fc2Context context,
                fc2Format7ImageSettings * imageSettings,
                unsigned int * packetSize,
                float * percentage )
```

Get the current Format7 configuration from the camera.

This call will only succeed if the camera is already in Format7.

Parameters

context	The fc2Context to be used.
imageSettings	Current image settings.
packetSize	Current packet size.
percentage	Current packet size as a percentage.

Returns

A fc2Error indicating the success or failure of the function.

6.11.2.2 fc2GetFormat7Info()

```
FLYCAPTURE2_C_API fc2Error fc2GetFormat7Info (
            fc2Context context,
            fc2Format7Info * info,
            BOOL * pSupported )
```

Retrieve the availability of Format7 custom image mode and the camera capabilities for the specified Format7 mode.

The mode must be specified in the Format7Info structure in order for the function to succeed.

Parameters

context	The fc2Context to be used.
info	Structure to be filled with the capabilities of the specified mode and the current state in the specified mode.
pSupported	Whether the specified mode is supported.

Returns

A fc2Error indicating the success or failure of the function.

6.11.2.3 fc2SetFormat7Configuration()

```
FLYCAPTURE2_C_API fc2Error fc2SetFormat7Configuration (
            fc2Context context,
            fc2Format7ImageSettings * imageSettings,
            float percentSpeed )
```

Set the current Format7 configuration to the camera.

Parameters

context	The fc2Context to be used.
imageSettings	Image settings to be written to the camera.
percentSpeed	Packet size as a percentage to be written to the camera.

Returns

A fc2Error indicating the success or failure of the function.

6.11.2.4 fc2SetFormat7ConfigurationPacket()

```
FLYCAPTURE2_C_API fc2Error fc2SetFormat7ConfigurationPacket (
            fc2Context context,
            fc2Format7ImageSettings * imageSettings,
            unsigned int packetSize )
```

Set the current Format7 configuration to the camera.

Parameters

context	The fc2Context to be used.
imageSettings	Image settings to be written to the camera.
packetSize	Packet size to be written to the camera.

Returns

A fc2Error indicating the success or failure of the function.

6.11.2.5 fc2ValidateFormat7Settings()

```
FLYCAPTURE2_C_API fc2Error fc2ValidateFormat7Settings (
            fc2Context context,
            fc2Format7ImageSettings * imageSettings,
            BOOL * settingsAreValid,
            fc2Format7PacketInfo * packetInfo )
```

Validates Format7ImageSettings structure and returns valid packet size information if the image settings are valid.

The current image settings are cached while validation is taking place. The cached settings are restored when validation is complete.

Parameters

context	The fc2Context to be used.
imageSettings	Structure containing the image settings.
settingsAreValid	Whether the settings are valid.
packetInfo	Packet size information that can be used to determine a valid packet size.

Returns

A fc2Error indicating the success or failure of the function.

6.12 GVCP Register Operation

These functions deal with GVCP register operation on the camera.

Functions

- FLYCAPTURE2_C_API fc2Error fc2WriteGVCPRegister (fc2Context context, unsigned int address, unsigned int value)

  Write a GVCP register.
- FLYCAPTURE2_C_API fc2Error fc2WriteGVCPRegisterBroadcast (fc2Context context, unsigned int address, unsigned int value)

  Write a GVCP register with broadcast.
- FLYCAPTURE2_C_API fc2Error fc2ReadGVCPRegister (fc2Context context, unsigned int address, unsigned int ∗pValue)

  Read a GVCP register.
- FLYCAPTURE2_C_API fc2Error fc2WriteGVCPRegisterBlock (fc2Context context, unsigned int address, const unsigned int ∗pBuffer, unsigned int length)

  Write a GVCP register block.
- FLYCAPTURE2_C_API fc2Error fc2ReadGVCPRegisterBlock (fc2Context context, unsigned int address, unsigned int ∗pBuffer, unsigned int length)

  Read a GVCP register block.
- FLYCAPTURE2_C_API fc2Error fc2WriteGVCPMemory (fc2Context context, unsigned int address, const unsigned char ∗pBuffer, unsigned int length)

  Write a GVCP memory block.
- FLYCAPTURE2_C_API fc2Error fc2ReadGVCPMemory (fc2Context context, unsigned int address, unsigned char ∗pBuffer, unsigned int length)

  Read a GVCP memory block.

6.12.1 Detailed Description

These functions deal with GVCP register operation on the camera.

6.12.2 Function Documentation

6.12.2.1 fc2ReadGVCPMemory()

```
FLYCAPTURE2_C_API fc2Error fc2ReadGVCPMemory (
             fc2Context context,
             unsigned int address,
             unsigned char ∗ pBuffer,
             unsigned int length )
```

Read a GVCP memory block.

Parameters

context	The fc2Context to be used.
address	GVCP address to be read from.
pBuffer	Array containing data to be written.
length	Size of array, in quadlets.

Returns

An Error indicating the success or failure of the function.

6.12.2.2 fc2ReadGVCPRegister()

```
FLYCAPTURE2_C_API fc2Error fc2ReadGVCPRegister (
             fc2Context context,
             unsigned int address,
             unsigned int * pValue )
```

Read a GVCP register.

Parameters

context	The fc2Context to be used.
address	GVCP address to be read from.
pValue	The value that is read.

Returns

An Error indicating the success or failure of the function.

6.12.2.3 fc2ReadGVCPRegisterBlock()

```
FLYCAPTURE2_C_API fc2Error fc2ReadGVCPRegisterBlock (
             fc2Context context,
             unsigned int address,
             unsigned int * pBuffer,
             unsigned int length )
```

Read a GVCP register block.

Parameters

context	The fc2Context to be used.
address	GVCP address to be read from.
pBuffer	Array containing data to be written.
length	Size of array, in quadlets.

Returns

An Error indicating the success or failure of the function.

6.12.2.4 fc2WriteGVCPMemory()

```
FLYCAPTURE2_C_API fc2Error fc2WriteGVCPMemory (
            fc2Context context,
            unsigned int address,
            const unsigned char * pBuffer,
            unsigned int length )
```

Write a GVCP memory block.

Parameters

context	The fc2Context to be used.
address	GVCP address to be write to.
pBuffer	Array containing data to be written.
length	Size of array, in quadlets.

Returns

An Error indicating the success or failure of the function.

6.12.2.5 fc2WriteGVCPRegister()

```
FLYCAPTURE2_C_API fc2Error fc2WriteGVCPRegister (
            fc2Context context,
            unsigned int address,
            unsigned int value )
```

Write a GVCP register.

Parameters

context	The fc2Context to be used.
address	GVCP address to be written to.
value	The value to be written.

Returns

An Error indicating the success or failure of the function.

6.12.2.6 fc2WriteGVCPRegisterBlock()

```
FLYCAPTURE2_C_API fc2Error fc2WriteGVCPRegisterBlock (
             fc2Context context,
             unsigned int address,
             const unsigned int * pBuffer,
             unsigned int length )
```

Write a GVCP register block.

Parameters

context	The fc2Context to be used.
address	GVCP address to be write to.
pBuffer	Array containing data to be written.
length	Size of array, in quadlets.

Returns

An Error indicating the success or failure of the function.

6.12.2.7 fc2WriteGVCPRegisterBroadcast()

```
FLYCAPTURE2_C_API fc2Error fc2WriteGVCPRegisterBroadcast (
             fc2Context context,
             unsigned int address,
             unsigned int value )
```

Write a GVCP register with broadcast.

Parameters

context	The fc2Context to be used.
address	GVCP address to be written to.
value	The value to be written.

Returns

An Error indicating the success or failure of the function.

6.13 GigE property manipulation

These functions deal with GigE properties.

Functions

- FLYCAPTURE2_C_API fc2Error fc2GetGigEProperty (fc2Context context, fc2GigEProperty ∗pGigEProp)

    Get the specified GigEProperty.
- FLYCAPTURE2_C_API fc2Error fc2SetGigEProperty (fc2Context context, const fc2GigEProperty ∗pGigE↩ Prop)

    Set the specified GigEProperty.
- FLYCAPTURE2_C_API fc2Error fc2DiscoverGigEPacketSize (fc2Context context, unsigned int ∗packetSize)

    Discover the largest packet size that works for the network link between the PC and the camera.

6.13.1 Detailed Description

These functions deal with GigE properties.

6.13.2 Function Documentation

6.13.2.1 fc2DiscoverGigEPacketSize()

```
FLYCAPTURE2_C_API fc2Error fc2DiscoverGigEPacketSize (
            fc2Context context,
            unsigned int * packetSize )
```

Discover the largest packet size that works for the network link between the PC and the camera.

This is useful in cases where there may be multiple links between the PC and the camera and there is a possibility of a component not supporting the recommended jumbo frame packet size of 9000.

Parameters

context	The fc2Context to be used.
packetSize	The maximum packet size supported by the link.

Returns

An Error indicating the success or failure of the function.

6.13.2.2 fc2GetGigEProperty()

```
FLYCAPTURE2_C_API fc2Error fc2GetGigEProperty (
             fc2Context context,
             fc2GigEProperty * pGigEProp )
```

Get the specified GigEProperty.

The GigEPropertyType field must be set in order for this function to succeed.

Parameters

context	The fc2Context to be used.
pGigEProp	The GigE property to get.

Returns

An Error indicating the success or failure of the function.

6.13.2.3 fc2SetGigEProperty()

```
FLYCAPTURE2_C_API fc2Error fc2SetGigEProperty (
             fc2Context context,
             const fc2GigEProperty * pGigEProp )
```

Set the specified GigEProperty.

The GigEPropertyType field must be set in order for this function to succeed.

Parameters

context	The fc2Context to be used.
pGigEProp	The GigE property to set.

Returns

An Error indicating the success or failure of the function.

6.14 GigE image settings

These functions deal with GigE image setting.

Functions

- FLYCAPTURE2_C_API fc2Error fc2QueryGigEImagingMode (fc2Context context, fc2Mode mode, BOOL ∗isSupported)

  Check if the particular imaging mode is supported by the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetGigEImagingMode (fc2Context context, fc2Mode ∗mode)

  Get the current imaging mode on the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetGigEImagingMode (fc2Context context, fc2Mode mode)

  Set the current imaging mode to the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetGigEImageSettingsInfo (fc2Context context, fc2GigEImage↩ SettingsInfo ∗pInfo)

  Get information about the image settings possible on the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetGigEImageSettings (fc2Context context, fc2GigEImageSettings ∗p↩ ImageSettings)

  Get the current image settings on the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetGigEImageSettings (fc2Context context, const fc2GigEImageSettings ∗pImageSettings)

  Set the image settings specified to the camera.

6.14.1 Detailed Description

These functions deal with GigE image setting.

6.14.2 Function Documentation

6.14.2.1 fc2GetGigEImageSettings()

```
FLYCAPTURE2_C_API fc2Error fc2GetGigEImageSettings (
            fc2Context context,
            fc2GigEImageSettings ∗ pImageSettings )
```

Get the current image settings on the camera.

Parameters

context	The fc2Context to be used.
pImageSettings	Current image settings on camera.

Returns

An Error indicating the success or failure of the function.

6.14.2.2 fc2GetGigEImageSettingsInfo()

```
FLYCAPTURE2_C_API fc2Error fc2GetGigEImageSettingsInfo (
            fc2Context context,
            fc2GigEImageSettingsInfo * pInfo )
```

Get information about the image settings possible on the camera.

Parameters

context	The fc2Context to be used.
pInfo	Image settings information.

Returns

An Error indicating the success or failure of the function.

6.14.2.3 fc2GetGigEImagingMode()

```
FLYCAPTURE2_C_API fc2Error fc2GetGigEImagingMode (
            fc2Context context,
            fc2Mode * mode )
```

Get the current imaging mode on the camera.

Parameters

context	The fc2Context to be used.
mode	Current imaging mode on the camera.

Returns

An Error indicating the success or failure of the function.

6.14.2.4 fc2QueryGigEImagingMode()

```
FLYCAPTURE2_C_API fc2Error fc2QueryGigEImagingMode (
            fc2Context context,
            fc2Mode mode,
            BOOL * isSupported )
```

Check if the particular imaging mode is supported by the camera.

Parameters

context	The fc2Context to be used.
mode	The mode to check.
isSupported	Whether the mode is supported.

Returns

An Error indicating the success or failure of the function.

6.14.2.5 fc2SetGigEImageSettings()

```
FLYCAPTURE2_C_API fc2Error fc2SetGigEImageSettings (
             fc2Context context,
             const fc2GigEImageSettings * pImageSettings )
```

Set the image settings specified to the camera.

Parameters

context	The fc2Context to be used.
pImageSettings	Image settings to set to camera.

Returns

An Error indicating the success or failure of the function.

6.14.2.6 fc2SetGigEImagingMode()

```
FLYCAPTURE2_C_API fc2Error fc2SetGigEImagingMode (
             fc2Context context,
             fc2Mode mode )
```

Set the current imaging mode to the camera.

This should only be done when the camera is not streaming images.

Parameters

context	The fc2Context to be used.
mode	Imaging mode to set to the camera.

Returns

An Error indicating the success or failure of the function.

6.15 GigE image binning settings

These functions deal with GigE image binning settings.

Functions

- FLYCAPTURE2_C_API fc2Error fc2GetGigEImageBinningSettings (fc2Context context, unsigned int ∗horz↩ BinnningValue, unsigned int ∗vertBinnningValue)

    Get the current binning settings on the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetGigEImageBinningSettings (fc2Context context, unsigned int horz↩ BinnningValue, unsigned int vertBinnningValue)

    Set the specified binning values to the camera.

6.15.1 Detailed Description

These functions deal with GigE image binning settings.

6.15.2 Function Documentation

6.15.2.1 fc2GetGigEImageBinningSettings()

```
FLYCAPTURE2_C_API fc2Error fc2GetGigEImageBinningSettings (
            fc2Context context,
            unsigned int * horzBinnningValue,
            unsigned int * vertBinnningValue )
```

Get the current binning settings on the camera.

Parameters

context	The fc2Context to be used.
horzBinnningValue	Current horizontal binning value.
vertBinnningValue	Current vertical binning value.

Returns

An Error indicating the success or failure of the function.

6.15.2.2 fc2SetGigEImageBinningSettings()

```
FLYCAPTURE2_C_API fc2Error fc2SetGigEImageBinningSettings (
            fc2Context context,
            unsigned int horzBinnningValue,
            unsigned int vertBinnningValue )
```

Set the specified binning values to the camera.

It is recommended that GetGigEImageSettingsInfo() be called after this function succeeds to retrieve the new image settings information for the new binning mode.

Parameters

context	The fc2Context to be used.
horzBinnningValue	Horizontal binning value.
vertBinnningValue	Vertical binning value.

Returns

An Error indicating the success or failure of the function.

6.16 GigE image stream configuration

These functions deal with GigE image stream configuration.

Functions

- FLYCAPTURE2_C_API fc2Error fc2GetNumStreamChannels (fc2Context context, unsigned int ∗num←
  Channels)

  Get the number of stream channels present on the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetGigEStreamChannelInfo (fc2Context context, unsigned int channel, fc2GigEStreamChannel ∗pChannel)

  Get the stream channel information for the specified channel.
- FLYCAPTURE2_C_API fc2Error fc2SetGigEStreamChannelInfo (fc2Context context, unsigned int channel, fc2GigEStreamChannel ∗pChannel)

  Set the stream channel information for the specified channel.
- FLYCAPTURE2_C_API fc2Error fc2GetGigEConfig (fc2Context context, fc2GigEConfig ∗pConfig)

  Get the current gige config on the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetGigEConfig (fc2Context context, const fc2GigEConfig ∗pConfig)

  Set the gige config specified to the camera.

6.16.1 Detailed Description

These functions deal with GigE image stream configuration.

6.16.2 Function Documentation

6.16.2.1 fc2GetGigEConfig()

```
FLYCAPTURE2_C_API fc2Error fc2GetGigEConfig (
            fc2Context context,
            fc2GigEConfig ∗ pConfig )
```

Get the current gige config on the camera.

Parameters

context	The fc2Context to be used.
pGigEConfig	Current configuration on camera.

Returns

An Error indicating the success or failure of the function.

6.16.2.2 fc2GetGigEStreamChannelInfo()

```
FLYCAPTURE2_C_API fc2Error fc2GetGigEStreamChannelInfo (
            fc2Context context,
            unsigned int channel,
            fc2GigEStreamChannel * pChannel )
```

Get the stream channel information for the specified channel.

Parameters

context	The fc2Context to be used.
channel	Channel number to use.
pChannel	Stream channel information for the specified channel.

Returns

An Error indicating the success or failure of the function.

6.16.2.3 fc2GetNumStreamChannels()

```
FLYCAPTURE2_C_API fc2Error fc2GetNumStreamChannels (
            fc2Context context,
            unsigned int * numChannels )
```

Get the number of stream channels present on the camera.

Parameters

context	The fc2Context to be used.
numChannels	Number of stream channels present.

Returns

An Error indicating the success or failure of the function.

6.16.2.4 fc2SetGigEConfig()

```
FLYCAPTURE2_C_API fc2Error fc2SetGigEConfig (
            fc2Context context,
            const fc2GigEConfig * pConfig )
```

Set the gige config specified to the camera.

Parameters

context	The fc2Context to be used.
pGigEConfig	configuration to set to camera.

Returns

An Error indicating the success or failure of the function.

6.16.2.5 fc2SetGigEStreamChannelInfo()

```
FLYCAPTURE2_C_API fc2Error fc2SetGigEStreamChannelInfo (
             fc2Context context,
             unsigned int channel,
             fc2GigEStreamChannel * pChannel )
```

Set the stream channel information for the specified channel.

Note that the source UDP port of the stream channel is read-only.

Parameters

context	The fc2Context to be used.
channel	Channel number to use.
pChannel	Stream channel information to use for the specified channel.

Returns

An Error indicating the success or failure of the function.

6.17 Image Operation

The Image operations are used to retrieve images from a camera, convert between multiple pixel formats and save images to disk.

Functions

- FLYCAPTURE2_C_API fc2Error fc2SetDefaultColorProcessing (fc2ColorProcessingAlgorithm default↩Method)

  Set the default color processing algorithm.
- FLYCAPTURE2_C_API fc2Error fc2GetDefaultColorProcessing (fc2ColorProcessingAlgorithm ∗pDefault↩Method)

  Get the default color processing algorithm.
- FLYCAPTURE2_C_API fc2Error fc2SetDefaultOutputFormat (fc2PixelFormat format)

  Set the default output pixel format.
- FLYCAPTURE2_C_API fc2Error fc2GetDefaultOutputFormat (fc2PixelFormat ∗pFormat)

  Get the default output pixel format.
- FLYCAPTURE2_C_API fc2Error fc2DetermineBitsPerPixel (fc2PixelFormat format, unsigned int ∗pBitsPer↩Pixel)

  Calculate the bits per pixel for the specified pixel format.
- FLYCAPTURE2_C_API fc2Error fc2CreateImage (fc2Image ∗pImage)

  Create a fc2Image.
- FLYCAPTURE2_C_API fc2Error fc2DestroyImage (fc2Image ∗image)

  Destroy the fc2Image.
- FLYCAPTURE2_C_API fc2Error fc2SetImageDimensions (fc2Image ∗pImage, unsigned int rows, unsigned int cols, unsigned int stride, fc2PixelFormat pixelFormat, fc2BayerTileFormat bayerFormat)

  Sets the dimensions of the image object.
- FLYCAPTURE2_C_API fc2Error fc2GetImageDimensions (fc2Image ∗pImage, unsigned int ∗pRows, unsigned int ∗pCols, unsigned int ∗pStride, fc2PixelFormat ∗pPixelFormat, fc2BayerTileFormat ∗pBayerFormat)

  Get the image dimensions associated with the image object.
- FLYCAPTURE2_C_API fc2Error fc2SetImageColorProcessing (fc2Image ∗pImage, fc2ColorProcessing↩Algorithm colorProc)

  Set the color processing algorithm.
- FLYCAPTURE2_C_API fc2Error fc2GetImageColorProcessing (fc2Image ∗pImage, fc2ColorProcessing↩Algorithm ∗pColorProc)

  Get the current color processing algorithm.
- FLYCAPTURE2_C_API fc2Error fc2SetImageData (fc2Image ∗pImage, const unsigned char ∗pData, unsigned int dataSize)

  Set the data of the Image object.
- FLYCAPTURE2_C_API fc2Error fc2GetImageData (fc2Image ∗pImage, unsigned char ∗∗ppData)

  Get a pointer to the data associated with the image.
- FLYCAPTURE2_C_API fc2Error fc2GetImageMetadata (fc2Image ∗pImage, fc2ImageMetadata ∗pImage↩MetaData)

  Get the metadata associated with the image.
- FLYCAPTURE2_C_API fc2TimeStamp fc2GetImageTimeStamp (fc2Image ∗pImage)

  Get the timestamp data associated with the image.
- FLYCAPTURE2_C_API fc2Error fc2SaveImage (fc2Image ∗pImage, const char ∗pFilename, fc2ImageFile↩Format format)

  Save the image to the specified file name with the file format specified.
- FLYCAPTURE2_C_API fc2Error fc2SaveImageWithOption (fc2Image ∗pImage, const char ∗pFilename, fc2ImageFileFormat format, void ∗pOption)

Save the image to the specified file name with the file format specified.
- FLYCAPTURE2_C_API fc2Error fc2ConvertImage (fc2Image ∗pImageIn, fc2Image ∗pImageOut)
- FLYCAPTURE2_C_API fc2Error fc2ConvertImageTo (fc2PixelFormat format, fc2Image ∗pImageIn, fc2Image ∗pImageOut)

  Converts the current image buffer to the specified output format and stores the result in the specified image.
- FLYCAPTURE2_C_API fc2Error fc2CalculateImageStatistics (fc2Image ∗pImage, fc2ImageStatisticsContext ∗pImageStatisticsContext)

  Calculate statistics associated with the image.

6.17.1 Detailed Description

The Image operations are used to retrieve images from a camera, convert between multiple pixel formats and save images to disk.

Operations on images are not guaranteed to be thread safe. It is recommended that operations on images be protected by thread synchronization constructs such as mutexes.

6.17.2 Function Documentation

6.17.2.1 fc2CalculateImageStatistics()

```
FLYCAPTURE2_C_API fc2Error fc2CalculateImageStatistics (
            fc2Image ∗ pImage,
            fc2ImageStatisticsContext ∗ pImageStatisticsContext )
```

Calculate statistics associated with the image.

In order to collect statistics for a particular channel, the enabled flag for the channel must be set to true. Statistics can only be collected for images in Mono8, Mono16, RGB, RGBU, BGR and BGRU.

Parameters

pImage	The fc2Image to be used.
pImageStatisticsContext	The fc2ImageStatisticsContext to hold the statistics.

Returns

A fc2Error indicating the success or failure of the function.

6.17.2.2 fc2ConvertImage()

```
FLYCAPTURE2_C_API fc2Error fc2ConvertImage (
            fc2Image ∗ pImageIn,
            fc2Image ∗ pImageOut )
```

Parameters

pImageIn	
pImageOut	

Returns

A fc2Error indicating the success or failure of the function.

6.17.2.3 fc2ConvertImageTo()

```
FLYCAPTURE2_C_API fc2Error fc2ConvertImageTo (
              fc2PixelFormat format,
              fc2Image * pImageIn,
              fc2Image * pImageOut )
```

Converts the current image buffer to the specified output format and stores the result in the specified image.

The destination image does not need to be configured in any way before the call is made.

Parameters

format	Output format of the converted image.
pImageIn	Input image.
pImageOut	Output image.

Returns

A fc2Error indicating the success or failure of the function.

6.17.2.4 fc2CreateImage()

```
FLYCAPTURE2_C_API fc2Error fc2CreateImage (
              fc2Image * pImage )
```

Create a fc2Image.

If externally allocated memory is to be used for the converted image, simply assigning the pData member of the fc2Image structure is insufficient. fc2SetImageData() should be called in order to populate the fc2Image structure correctly.

See also

fc2SetImageData()

Parameters

pImage	Pointer to image to be created.

Returns

A fc2Error indicating the success or failure of the function.

6.17.2.5 fc2DestroyImage()

```
FLYCAPTURE2_C_API fc2Error fc2DestroyImage (
             fc2Image * image )
```

Destroy the fc2Image.

Parameters

image	Pointer to image to be destroyed.

Returns

A fc2Error indicating the success or failure of the function.

6.17.2.6 fc2DetermineBitsPerPixel()

```
FLYCAPTURE2_C_API fc2Error fc2DetermineBitsPerPixel (
             fc2PixelFormat format,
             unsigned int * pBitsPerPixel )
```

Calculate the bits per pixel for the specified pixel format.

Parameters

format	The pixel format.
pBitsPerPixel	The bits per pixel.

Returns

A fc2Error indicating the success or failure of the function.

6.17.2.7 fc2GetDefaultColorProcessing()

```
FLYCAPTURE2_C_API fc2Error fc2GetDefaultColorProcessing (
            fc2ColorProcessingAlgorithm * pDefaultMethod )
```

Get the default color processing algorithm.

Parameters

pDefaultMethod	The default color processing algorithm.

Returns

A fc2Error indicating the success or failure of the function.

6.17.2.8 fc2GetDefaultOutputFormat()

```
FLYCAPTURE2_C_API fc2Error fc2GetDefaultOutputFormat (
            fc2PixelFormat * pFormat )
```

Get the default output pixel format.

Parameters

pFormat	The default pixel format.

Returns

A fc2Error indicating the success or failure of the function.

6.17.2.9 fc2GetImageColorProcessing()

```
FLYCAPTURE2_C_API fc2Error fc2GetImageColorProcessing (
            fc2Image * pImage,
            fc2ColorProcessingAlgorithm * pColorProc )
```

Get the current color processing algorithm.

Parameters

pImage	The fc2Image to be used.

See also

fc2SetColorProcessing()

Returns

The current color processing algorithm.

6.17.2.10 fc2GetImageData()

```
FLYCAPTURE2_C_API fc2Error fc2GetImageData (
             fc2Image * pImage,
             unsigned char ** ppData )
```

Get a pointer to the data associated with the image.

This function is considered unsafe. The pointer returned could be invalidated if the buffer is resized or released. The pointer may also be invalidated if the Image object is passed to fc2RetrieveBuffer().

Parameters

pImage	The fc2Image to be used.
ppData	A pointer to the image data.

Returns

A fc2Error indicating the success or failure of the function.

6.17.2.11 fc2GetImageDimensions()

```
FLYCAPTURE2_C_API fc2Error fc2GetImageDimensions (
             fc2Image * pImage,
             unsigned int * pRows,
             unsigned int * pCols,
             unsigned int * pStride,
             fc2PixelFormat * pPixelFormat,
             fc2BayerTileFormat * pBayerFormat )
```

Get the image dimensions associated with the image object.

Parameters

pImage	The fc2Image to be used.
pRows	Number of rows.
pCols	Number of columns.
pStride	The stride.
pPixelFormat	Pixel format.
pBayerFormat	Bayer tile format.

6.17.2.12 fc2GetImageMetadata()

```
FLYCAPTURE2_C_API fc2Error fc2GetImageMetadata (
             fc2Image * pImage,
             fc2ImageMetadata * pImageMetaData )
```

Get the metadata associated with the image.

This includes embedded image information.

Parameters

pImage	The fc2Image to be used.

Returns

Metadata associated with the image.

6.17.2.13 fc2GetImageTimeStamp()

```
FLYCAPTURE2_C_API fc2TimeStamp fc2GetImageTimeStamp (
             fc2Image * pImage )
```

Get the timestamp data associated with the image.

Parameters

pImage	The fc2Image to be used.

Returns

Timestamp data associated with the image.

6.17.2.14 fc2SaveImage()

```
FLYCAPTURE2_C_API fc2Error fc2SaveImage (
             fc2Image * pImage,
             const char * pFilename,
             fc2ImageFileFormat format )
```

Save the image to the specified file name with the file format specified.

Parameters

pImage	The fc2Image to be used.
pFilename	Filename to save image with.
format	File format to save in.

Returns

A fc2Error indicating the success or failure of the function.

6.17.2.15 fc2SaveImageWithOption()

```
FLYCAPTURE2_C_API fc2Error fc2SaveImageWithOption (
             fc2Image * pImage,
             const char * pFilename,
             fc2ImageFileFormat format,
             void * pOption )
```

Save the image to the specified file name with the file format specified.

Parameters

pImage	The fc2Image to be used.
pFilename	Filename to save image with.
format	File format to save in.
pOption	Options for saving image.

Returns

A fc2Error indicating the success or failure of the function.

6.17.2.16 fc2SetDefaultColorProcessing()

```
FLYCAPTURE2_C_API fc2Error fc2SetDefaultColorProcessing (
             fc2ColorProcessingAlgorithm defaultMethod )
```

Set the default color processing algorithm.

This method will be used for any image with the DEFAULT algorithm set. The method used is determined at the time of the Convert() call, therefore the most recent execution of this function will take precedence. The default setting is shared within the current process.

Parameters

defaultMethod	The color processing algorithm to set.

Returns

A fc2Error indicating the success or failure of the function.

6.17.2.17 fc2SetDefaultOutputFormat()

```
FLYCAPTURE2_C_API fc2Error fc2SetDefaultOutputFormat (
              fc2PixelFormat format )
```

Set the default output pixel format.

This format will be used for any call to Convert() that does not specify an output format. The format used will be determined at the time of the Convert() call, therefore the most recent execution of this function will take precedence. The default is shared within the current process.

Parameters

format	The output pixel format to set.

Returns

A fc2Error indicating the success or failure of the function.

6.17.2.18 fc2SetImageColorProcessing()

```
FLYCAPTURE2_C_API fc2Error fc2SetImageColorProcessing (
              fc2Image * pImage,
              fc2ColorProcessingAlgorithm colorProc )
```

Set the color processing algorithm.

This should be set on the input image object.

Parameters

pImage	The fc2Image to be used.
colorProc	The color processing algorithm to use.

See also

fc2GetColorProcessing()

Returns

An Error indicating the success or failure of the function.

6.17.2.19 fc2SetImageData()

```
FLYCAPTURE2_C_API fc2Error fc2SetImageData (
            fc2Image * pImage,
            const unsigned char * pData,
            unsigned int dataSize )
```

Set the data of the Image object.

Ownership of the image buffer is not transferred to the Image object. It is the user's responsibility to delete the buffer when it is no longer in use.

Parameters

pImage	The fc2Image to be used.
pData	Pointer to the image buffer.
dataSize	Size of the image buffer.

Returns

A fc2Error indicating the success or failure of the function.

6.17.2.20 fc2SetImageDimensions()

```
FLYCAPTURE2_C_API fc2Error fc2SetImageDimensions (
            fc2Image * pImage,
            unsigned int rows,
            unsigned int cols,
            unsigned int stride,
            fc2PixelFormat pixelFormat,
            fc2BayerTileFormat bayerFormat )
```

Sets the dimensions of the image object.

Parameters

pImage	The fc2Image to be used.
rows	Number of rows to set.
cols	Number of cols to set.
stride	Stride to set.
pixelFormat	Pixel format to set.
bayerFormat	Bayer tile format to set.

Returns

A fc2Error indicating the success or failure of the function.

6.18 Image Statistics Operation

The Image Statistics operation provides the functionality for the user to collect image channel statistics.

Functions

- FLYCAPTURE2_C_API fc2Error fc2CreateImageStatistics (fc2ImageStatisticsContext ∗pImageStatistics↩Context)

  Create a statistics context.
- FLYCAPTURE2_C_API fc2Error fc2DestroyImageStatistics (fc2ImageStatisticsContext imageStatistics↩Context)

  Destroy a statistics context.
- FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableAll (fc2ImageStatisticsContext imageStatistics↩Context)

  Enable all channels.
- FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsDisableAll (fc2ImageStatisticsContext imageStatistics↩Context)

  Disable all channels.
- FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableGreyOnly (fc2ImageStatisticsContext image↩StatisticsContext)

  Enable only the grey channel.
- FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableRGBOnly (fc2ImageStatisticsContext image↩StatisticsContext)

  Enable only the RGB channels.
- FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableHSLOnly (fc2ImageStatisticsContext image↩StatisticsContext)

  Enable only the HSL channels.
- FLYCAPTURE2_C_API fc2Error fc2GetChannelStatus (fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, BOOL ∗pEnabled)

  Get the status of a statistics channel.
- FLYCAPTURE2_C_API fc2Error fc2SetChannelStatus (fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, BOOL enabled)

  Set the status of a statistics channel.
- FLYCAPTURE2_C_API fc2Error fc2GetChannelRange (fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, unsigned int ∗pMin, unsigned int ∗pMax)

  Get the range of a statistics channel.
- FLYCAPTURE2_C_API fc2Error fc2GetChannelPixelValueRange (fc2ImageStatisticsContext image↩StatisticsContext, fc2StatisticsChannel channel, unsigned int ∗pPixelValueMin, unsigned int ∗pPixel↩ValueMax)

  Get the range of a statistics channel.
- FLYCAPTURE2_C_API fc2Error fc2GetChannelNumPixelValues (fc2ImageStatisticsContext image↩StatisticsContext, fc2StatisticsChannel channel, unsigned int ∗pNumPixelValues)

  Get the number of unique pixel values in the image.
- FLYCAPTURE2_C_API fc2Error fc2GetChannelMean (fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, float ∗pPixelValueMean)

  Get the mean of the image.
- FLYCAPTURE2_C_API fc2Error fc2GetChannelHistogram (fc2ImageStatisticsContext imageStatistics↩Context, fc2StatisticsChannel channel, int ∗∗ppHistogram)

  Get the histogram for the image.
- FLYCAPTURE2_C_API fc2Error fc2GetImageStatistics (fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, unsigned int ∗pRangeMin, unsigned int ∗pRangeMax, unsigned int ∗p↩PixelValueMin, unsigned int ∗pPixelValueMax, unsigned int ∗pNumPixelValues, float ∗pPixelValueMean, int ∗∗ppHistogram)

  Get all statistics for the image.

6.18.1 Detailed Description

The Image Statistics operation provides the functionality for the user to collect image channel statistics.

6.18.2 Function Documentation

6.18.2.1 fc2CreateImageStatistics()

```
FLYCAPTURE2_C_API fc2Error fc2CreateImageStatistics (
            fc2ImageStatisticsContext * pImageStatisticsContext )
```

Create a statistics context.

Parameters

pImageStatisticsContext	A statistics context.

Returns

A fc2Error indicating the success or failure of the function.

6.18.2.2 fc2DestroyImageStatistics()

```
FLYCAPTURE2_C_API fc2Error fc2DestroyImageStatistics (
            fc2ImageStatisticsContext imageStatisticsContext )
```

Destroy a statistics context.

Parameters

imageStatisticsContext	A statistics context.

Returns

A fc2Error indicating the success or failure of the function.

6.18.2.3 fc2GetChannelHistogram()

```
FLYCAPTURE2_C_API fc2Error fc2GetChannelHistogram (
            fc2ImageStatisticsContext imageStatisticsContext,
            fc2StatisticsChannel channel,
            int ** ppHistogram )
```

Get the histogram for the image.

Parameters

imageStatisticsContext	A statistics context.
channel	The statistics channel.
ppHistogram	Pointer to an array containing the histogram.

Returns

An Error indicating the success or failure of the function.

6.18.2.4 fc2GetChannelMean()

```
FLYCAPTURE2_C_API fc2Error fc2GetChannelMean (
            fc2ImageStatisticsContext imageStatisticsContext,
            fc2StatisticsChannel channel,
            float * pPixelValueMean )
```

Get the mean of the image.

Parameters

imageStatisticsContext	A statistics context.
channel	The statistics channel.
pPixelValueMean	The mean of the image.

Returns

An Error indicating the success or failure of the function.

6.18.2.5 fc2GetChannelNumPixelValues()

```
FLYCAPTURE2_C_API fc2Error fc2GetChannelNumPixelValues (
            fc2ImageStatisticsContext imageStatisticsContext,
            fc2StatisticsChannel channel,
            unsigned int * pNumPixelValues )
```

Get the number of unique pixel values in the image.

Parameters

imageStatisticsContext	A statistics context.
channel	The statistics channel.
pNumPixelValues	The number of unique pixel values.

Returns

An Error indicating the success or failure of the function.

6.18.2.6 fc2GetChannelPixelValueRange()

```
FLYCAPTURE2_C_API fc2Error fc2GetChannelPixelValueRange (
            fc2ImageStatisticsContext imageStatisticsContext,
            fc2StatisticsChannel channel,
            unsigned int * pPixelValueMin,
            unsigned int * pPixelValueMax )
```

Get the range of a statistics channel.

The values returned are the maximum values recorded for all pixels in the image.

Parameters

imageStatisticsContext	A statistics context.
channel	The statistics channel.
pPixelValueMin	The minimum pixel value.
pPixelValueMax	The maximum pixel value.

Returns

An Error indicating the success or failure of the function.

6.18.2.7 fc2GetChannelRange()

```
FLYCAPTURE2_C_API fc2Error fc2GetChannelRange (
            fc2ImageStatisticsContext imageStatisticsContext,
            fc2StatisticsChannel channel,
            unsigned int * pMin,
            unsigned int * pMax )
```

Get the range of a statistics channel.

The values returned are the maximum possible values for any given pixel in the image. This is generally 0-255 for 8 bit images, and 0-65535 for 16 bit images.

Parameters

imageStatisticsContext	A statistics context.
channel	The statistics channel.
pMin	The minimum possible value.
pMax	The maximum possible value.

Returns

An Error indicating the success or failure of the function.

6.18.2.8 fc2GetChannelStatus()

```
FLYCAPTURE2_C_API fc2Error fc2GetChannelStatus (
             fc2ImageStatisticsContext imageStatisticsContext,
             fc2StatisticsChannel channel,
             BOOL * pEnabled )
```

Get the status of a statistics channel.

See also

fc2SetChannelStatus()

Parameters

imageStatisticsContext	A statistics context.
channel	The statistics channel.
pEnabled	Whether the channel is enabled.

Returns

An Error indicating the success or failure of the function.

6.18.2.9 fc2GetImageStatistics()

```
FLYCAPTURE2_C_API fc2Error fc2GetImageStatistics (
             fc2ImageStatisticsContext imageStatisticsContext,
             fc2StatisticsChannel channel,
             unsigned int * pRangeMin,
             unsigned int * pRangeMax,
             unsigned int * pPixelValueMin,
             unsigned int * pPixelValueMax,
             unsigned int * pNumPixelValues,
             float * pPixelValueMean,
             int ** ppHistogram )
```

Get all statistics for the image.

Parameters

imageStatisticsContext	The statistics context.
channel	The statistics channel.
pRangeMin	The minimum possible value.

Parameters

pRangeMax	The maximum possible value.
pPixelValueMin	The minimum pixel value.
pPixelValueMax	The maximum pixel value.
pNumPixelValues	The number of unique pixel values.
pPixelValueMean	The mean of the image.
ppHistogram	Pointer to an array containing the histogram.

Returns

A fc2Error indicating the success or failure of the function.

6.18.2.10 fc2ImageStatisticsDisableAll()

```
FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsDisableAll (
            fc2ImageStatisticsContext imageStatisticsContext )
```

Disable all channels.

Parameters

imageStatisticsContext	A statistics context.

Returns

An Error indicating the success or failure of the function.

6.18.2.11 fc2ImageStatisticsEnableAll()

```
FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableAll (
            fc2ImageStatisticsContext imageStatisticsContext )
```

Enable all channels.

Parameters

imageStatisticsContext	A statistics context.

Returns

An Error indicating the success or failure of the function.

6.18.2.12 fc2ImageStatisticsEnableGreyOnly()

```
FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableGreyOnly (
            fc2ImageStatisticsContext imageStatisticsContext )
```

Enable only the grey channel.

Parameters

imageStatisticsContext	A statistics context.

Returns

An Error indicating the success or failure of the function.

6.18.2.13 fc2ImageStatisticsEnableHSLOnly()

```
FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableHSLOnly (
            fc2ImageStatisticsContext imageStatisticsContext )
```

Enable only the HSL channels.

Parameters

imageStatisticsContext	A statistics context.

Returns

An Error indicating the success or failure of the function.

6.18.2.14 fc2ImageStatisticsEnableRGBOnly()

```
FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableRGBOnly (
            fc2ImageStatisticsContext imageStatisticsContext )
```

Enable only the RGB channels.

Parameters

imageStatisticsContext	A statistics context.

Returns

An Error indicating the success or failure of the function.

6.18.2.15 fc2SetChannelStatus()

```
FLYCAPTURE2_C_API fc2Error fc2SetChannelStatus (
            fc2ImageStatisticsContext imageStatisticsContext,
            fc2StatisticsChannel channel,
            BOOL enabled )
```

Set the status of a statistics channel.

See also

fc2GetChannelStatus()

Parameters

imageStatisticsContext	A statistics context.
channel	The statistics channel.
enabled	Whether the channel should be enabled.

Returns

An Error indicating the success or failure of the function.

6.19 TopologyNode Operation

The TopologyNode operation provides the functionality for the user to generate a tree structure of all cameras and devices connected to a computer.

Functions

- FLYCAPTURE2_C_API fc2Error fc2CreateTopologyNode (fc2TopologyNodeContext ∗pTopologyNode↩ Context)

    Create a TopologyNode context.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetGuid (fc2TopologyNodeContext TopologyNode↩ Context, fc2PGRGuid ∗pGuid)

    Get the PGRGuid associated with the node.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetDeviceId (fc2TopologyNodeContext TopologyNode↩ Context, int ∗pID)

    Get the device ID associated with the node.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetNodeType (fc2TopologyNodeContext TopologyNode↩ Context, fc2NodeType ∗pNodeType)

    Get the node type associated with the node.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetInterfaceType (fc2TopologyNodeContext Topology↩ NodeContext, fc2InterfaceType ∗pInterfaceType)

    Get the interface type associated with the node.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetNumChildren (fc2TopologyNodeContext Topology↩ NodeContext, unsigned int ∗pNumChildNodes)

    Get the number of child nodes.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetChild (fc2TopologyNodeContext TopologyNode↩ Context, unsigned int position, fc2TopologyNodeContext ∗pChildTopologyNodeContext)

    Get child node located at the specified position.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeAddChild (fc2TopologyNodeContext TopologyNode↩ Context, fc2TopologyNodeContext TopologyNodeChildContext)

    Add the specified TopologyNode as a child of the node.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetNumPorts (fc2TopologyNodeContext TopologyNode↩ Context, unsigned int ∗pNumPorts)

    Get the number of ports.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetPortType (fc2TopologyNodeContext TopologyNode↩ Context, unsigned int position, fc2PortType ∗pPortType)

    Get type of port located at the specified position.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeAddPortType (fc2TopologyNodeContext TopologyNode↩ Context, fc2PortType portType)

    Add the specified PortType as a port of the node.
- FLYCAPTURE2_C_API BOOL fc2TopologyNodeAssignGuidToNode (fc2TopologyNodeContext Topology↩ NodeContext, fc2PGRGuid guid, int deviceId)

    Assign a PGRGuid and device ID to the node.
- FLYCAPTURE2_C_API BOOL fc2TopologyNodeAssignGuidToNodeEx (fc2TopologyNodeContext Topology↩ NodeContext, fc2PGRGuid guid, int deviceId, fc2NodeType nodeType)

    Assign a PGRGuid, device ID and nodeType to the node.
- FLYCAPTURE2_C_API fc2Error fc2DestroyTopologyNode (fc2TopologyNodeContext TopologyNodeContext)

    Destroy a TopologyNode context.

6.19.1 Detailed Description

The TopologyNode operation provides the functionality for the user to generate a tree structure of all cameras and devices connected to a computer.

6.19.2 Function Documentation

6.19.2.1 fc2CreateTopologyNode()

```
FLYCAPTURE2_C_API fc2Error fc2CreateTopologyNode (
             fc2TopologyNodeContext * pTopologyNodeContext )
```

Create a TopologyNode context.

Parameters

pTopologyNodeContext	A Topology Node context.

Returns

A fc2Error indicating the success or failure of the function.

6.19.2.2 fc2DestroyTopologyNode()

```
FLYCAPTURE2_C_API fc2Error fc2DestroyTopologyNode (
             fc2TopologyNodeContext TopologyNodeContext )
```

Destroy a TopologyNode context.

Parameters

TopologyNodeContext	A Topology Node context.

Returns

A fc2Error indicating the success or failure of the function.

6.19.2.3 fc2TopologyNodeAddChild()

```
FLYCAPTURE2_C_API fc2Error fc2TopologyNodeAddChild (
             fc2TopologyNodeContext TopologyNodeContext,
             fc2TopologyNodeContext TopologyNodeChildContext )
```

Add the specified TopologyNode as a child of the node.

Parameters

TopologyNodeContext	The Topology Node context to use.
TopologyNodeChildContext	The TopologyNode child context to add.

Returns

A fc2Error indicating the success or failure of the function.

6.19.2.4 fc2TopologyNodeAddPortType()

```
FLYCAPTURE2_C_API fc2Error fc2TopologyNodeAddPortType (
            fc2TopologyNodeContext TopologyNodeContext,
            fc2PortType portType )
```

Add the specified PortType as a port of the node.

Parameters

TopologyNodeContext	The Topology Node context to use.
portType	childPort The port to add.

Returns

A fc2Error indicating the success or failure of the function.

6.19.2.5 fc2TopologyNodeAssignGuidToNode()

```
FLYCAPTURE2_C_API BOOL fc2TopologyNodeAssignGuidToNode (
            fc2TopologyNodeContext TopologyNodeContext,
            fc2PGRGuid guid,
            int deviceId )
```

Assign a PGRGuid and device ID to the node.

Parameters

TopologyNodeContext	The Topology Node context to use.
guid	PGRGuid to be assigned.
deviceId	Device ID to be assigned.

Returns

A fc2Error indicating the success or failure of the function.

6.19.2.6 fc2TopologyNodeAssignGuidToNodeEx()

```
FLYCAPTURE2_C_API BOOL fc2TopologyNodeAssignGuidToNodeEx (
            fc2TopologyNodeContext TopologyNodeContext,
            fc2PGRGuid guid,
            int deviceId,
            fc2NodeType nodeType )
```

Assign a PGRGuid, device ID and nodeType to the node.

Parameters

TopologyNodeContext	The Topology Node context to use.
guid	PGRGuid to be assigned.
deviceId	Device ID to be assigned.
nodeType	NodeType to be assigned

Returns

A fc2Error indicating the success or failure of the function.

6.19.2.7 fc2TopologyNodeGetChild()

```
FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetChild (
            fc2TopologyNodeContext TopologyNodeContext,
            unsigned int position,
            fc2TopologyNodeContext * pChildTopologyNodeContext )
```

Get child node located at the specified position.

Parameters

TopologyNodeContext	The Topology Node context to use.
position	Position of the child node.
pChildTopologyNodeContext	The Topology Node context the contains information on the child topology

Returns

A fc2Error indicating the success or failure of the function.

6.19.2.8 fc2TopologyNodeGetDeviceId()

```
FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetDeviceId (
            fc2TopologyNodeContext TopologyNodeContext,
            int * pID )
```

Get the device ID associated with the node.

Parameters

TopologyNodeContext	The Topology Node context to use.
pID	Device ID of the node.

Returns

A fc2Error indicating the success or failure of the function.

6.19.2.9 fc2TopologyNodeGetGuid()

```
FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetGuid (
            fc2TopologyNodeContext TopologyNodeContext,
            fc2PGRGuid * pGuid )
```

Get the PGRGuid associated with the node.

Parameters

TopologyNodeContext	The Topology Node context to use.
pGuid	The unique identifier associated with the node.

Returns

A fc2Error indicating the success or failure of the function.

6.19.2.10 fc2TopologyNodeGetInterfaceType()

```
FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetInterfaceType (
            fc2TopologyNodeContext TopologyNodeContext,
            fc2InterfaceType * pInterfaceType )
```

Get the interface type associated with the node.

Parameters

TopologyNodeContext	The Topology Node context to use.
pInterfaceType	Interface type of the node.

Returns

A fc2Error indicating the success or failure of the function.

6.19.2.11 fc2TopologyNodeGetNodeType()

```
FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetNodeType (
            fc2TopologyNodeContext TopologyNodeContext,
            fc2NodeType * pNodeType )
```

Get the node type associated with the node.

Parameters

TopologyNodeContext	The Topology Node context to use.
pNodeType	Node type of the node.

Returns

A fc2Error indicating the success or failure of the function.

6.19.2.12 fc2TopologyNodeGetNumChildren()

```
FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetNumChildren (
            fc2TopologyNodeContext TopologyNodeContext,
            unsigned int * pNumChildNodes )
```

Get the number of child nodes.

Parameters

TopologyNodeContext	The Topology Node context to use.
pNumChildNodes	Number of child nodes.

Returns

A fc2Error indicating the success or failure of the function.

6.19.2.13 fc2TopologyNodeGetNumPorts()

```
FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetNumPorts (
            fc2TopologyNodeContext TopologyNodeContext,
            unsigned int * pNumPorts )
```

Get the number of ports.

Parameters

TopologyNodeContext	The Topology Node context to use.
pNumPorts	Number of ports.

Returns

A fc2Error indicating the success or failure of the function.

6.19.2.14 fc2TopologyNodeGetPortType()

```
FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetPortType (
            fc2TopologyNodeContext TopologyNodeContext,
            unsigned int position,
            fc2PortType * pPortType )
```

Get type of port located at the specified position.

Parameters

TopologyNodeContext	The Topology Node context to use.
position	Position of the port.
pPortType	PortType at the specified position.

Returns

A fc2Error indicating the success or failure of the function.

6.20 Utilities

The utility operations are used to query for general system information such as operating system, available memory etc.

Functions

- FLYCAPTURE2_C_API fc2Error fc2CheckDriver (const fc2PGRGuid ∗pGuid)

  Check for driver compatibility for the given camera guid.
- FLYCAPTURE2_C_API fc2Error fc2GetDriverDeviceName (const fc2PGRGuid ∗pGuid, char ∗pDevice←Name, size_t ∗deviceNameLength)

  Get the driver's name for a device.
- FLYCAPTURE2_C_API fc2Error fc2GetSystemInfo (fc2SystemInfo ∗pSystemInfo)

  Get system information.
- FLYCAPTURE2_C_API fc2Error fc2GetLibraryVersion (fc2Version ∗pVersion)

  Get library version.
- FLYCAPTURE2_C_API fc2Error fc2LaunchBrowser (const char ∗pAddress)

  Launch a URL in the system default browser.
- FLYCAPTURE2_C_API fc2Error fc2LaunchHelp (const char ∗pFileName)

  Open a CHM file in the system default CHM viewer.
- FLYCAPTURE2_C_API fc2Error fc2LaunchCommand (const char ∗pCommand)

  Execute a command in the terminal.
- FLYCAPTURE2_C_API fc2Error fc2LaunchCommandAsync (const char ∗pCommand, fc2AsyncCommand←Callback pCallback, void ∗pUserData)

  Execute a command in the terminal.
- FLYCAPTURE2_C_API const char ∗ fc2ErrorToDescription (fc2Error error)

  Get a string representation of an error.

6.20.1 Detailed Description

The utility operations are used to query for general system information such as operating system, available memory etc.

It can also be used to launch browsers, CHM viewers or terminal commands.

6.20.2 Function Documentation

6.20.2.1 fc2CheckDriver()

```
FLYCAPTURE2_C_API fc2Error fc2CheckDriver (
            const fc2PGRGuid ∗ pGuid )
```

Check for driver compatibility for the given camera guid.

Parameters

pGuid	The PGRGuid of the device to check.

Returns

FC2_ERROR_OK if the library is compatible with the currently loaded driver, otherwise an error indicating the type of failure.

6.20.2.2 fc2ErrorToDescription()

```
FLYCAPTURE2_C_API const char* fc2ErrorToDescription (
              fc2Error error )
```

Get a string representation of an error.

Parameters

error	Error to be parsed.

Returns

A fc2Error indicating the success or failure of the function.

6.20.2.3 fc2GetDriverDeviceName()

```
FLYCAPTURE2_C_API fc2Error fc2GetDriverDeviceName (
              const fc2PGRGuid * pGuid,
              char * pDeviceName,
              size_t * deviceNameLength )
```

Get the driver's name for a device.

Parameters

pGuid	The PGRGuid of the device to check.
pDeviceName	The device name will be returned in this string
pDeviceNameLength	The length of the device name string returned

Returns

An Error indicating the success or failure of the function.

6.20.2.4 fc2GetLibraryVersion()

```
FLYCAPTURE2_C_API fc2Error fc2GetLibraryVersion (
             fc2Version * pVersion )
```

Get library version.

Parameters

pVersion	Structure to receive the library version.

Returns

A fc2Error indicating the success or failure of the function.

6.20.2.5 fc2GetSystemInfo()

```
FLYCAPTURE2_C_API fc2Error fc2GetSystemInfo (
             fc2SystemInfo * pSystemInfo )
```

Get system information.

Parameters

pSystemInfo	Structure to receive system information.

Returns

A fc2Error indicating the success or failure of the function.

6.20.2.6 fc2LaunchBrowser()

```
FLYCAPTURE2_C_API fc2Error fc2LaunchBrowser (
             const char * pAddress )
```

Launch a URL in the system default browser.

Parameters

pAddress	URL to open in browser.

Returns

A fc2Error indicating the success or failure of the function.

6.20.2.7 fc2LaunchCommand()

```
FLYCAPTURE2_C_API fc2Error fc2LaunchCommand (
              const char * pCommand )
```

Execute a command in the terminal.

This is a blocking call that will return when the command completes.

Parameters

pCommand	Command to execute.

Returns

A fc2Error indicating the success or failure of the function.

6.20.2.8 fc2LaunchCommandAsync()

```
FLYCAPTURE2_C_API fc2Error fc2LaunchCommandAsync (
              const char * pCommand,
              fc2AsyncCommandCallback pCallback,
              void * pUserData )
```

Execute a command in the terminal.

This is a non-blocking call that will return immediately. The return value of the command can be retrieved in the callback.

Parameters

pCommand	Command to execute.
pCallback	Callback to fire when command is complete.
pUserData	Data pointer to pass to callback.

Returns

A fc2Error indicating the success or failure of the function.

6.20.2.9 fc2LaunchHelp()

```
FLYCAPTURE2_C_API fc2Error fc2LaunchHelp (
              const char * pFileName )
```

Open a CHM file in the system default CHM viewer.

Parameters

pFileName	Filename of CHM file to open.

Returns

A fc2Error indicating the success or failure of the function.

6.21 TypeDefs

Data Structures

- struct fc2PGRGuid

  A GUID to the camera.

Macros

- #define FALSE 0
- #define TRUE 1
- #define FULL_32BIT_VALUE 0x7FFFFFFF
- #define MAX_STRING_LENGTH 512

Typedefs

- typedef int BOOL
- typedef void ∗ fc2Context

  A context to the FlyCapture2 C library.
- typedef void ∗ fc2GuiContext

  A context to the FlyCapture2 C GUI library.
- typedef void ∗ fc2ImageImpl

  An internal pointer used in the fc2Image structure.
- typedef void ∗ fc2ImageStatisticsContext

  A context referring to the ImageStatistics object.
- typedef void ∗ fc2TopologyNodeContext

  A context referring to the TopologyNode object.
- typedef void ∗ fc2VideoContext

  A context referring to the video recorder object.

6.21.1 Detailed Description

6.21.2 Macro Definition Documentation

6.21.2.1 FALSE

```
#define FALSE 0
```

6.21.2.2 FULL_32BIT_VALUE

```
#define FULL_32BIT_VALUE 0x7FFFFFFF
```

6.21.2.3 MAX_STRING_LENGTH

```
#define MAX_STRING_LENGTH 512
```

6.21.2.4 TRUE

```
#define TRUE 1
```

6.21.3 Typedef Documentation

6.21.3.1 BOOL

```
typedef int BOOL
```

6.21.3.2 fc2Context

```
typedef void* fc2Context
```

A context to the FlyCapture2 C library.

It must be created before performing any calls to the library.

6.21.3.3 fc2GuiContext

```
typedef void* fc2GuiContext
```

A context to the FlyCapture2 C GUI library.

It must be created before performing any calls to the library.

6.21.3.4 fc2ImageImpl

```
typedef void* fc2ImageImpl
```

An internal pointer used in the fc2Image structure.

6.21.3.5 fc2ImageStatisticsContext

```
typedef void* fc2ImageStatisticsContext
```

A context referring to the ImageStatistics object.

6.21.3.6 fc2TopologyNodeContext

```
typedef void* fc2TopologyNodeContext
```

A context referring to the TopologyNode object.

6.21.3.7 fc2VideoContext

```
typedef void* fc2VideoContext
```

A context referring to the video recorder object.

6.22 Enumerations

Enumerations

- enum fc2Error {
  FC2_ERROR_UNDEFINED = -1,
  FC2_ERROR_OK,
  FC2_ERROR_FAILED,
  FC2_ERROR_NOT_IMPLEMENTED,
  FC2_ERROR_FAILED_BUS_MASTER_CONNECTION,
  FC2_ERROR_NOT_CONNECTED,
  FC2_ERROR_INIT_FAILED,
  FC2_ERROR_NOT_INTITIALIZED,
  FC2_ERROR_INVALID_PARAMETER,
  FC2_ERROR_INVALID_SETTINGS,
  FC2_ERROR_INVALID_BUS_MANAGER,
  FC2_ERROR_MEMORY_ALLOCATION_FAILED,
  FC2_ERROR_LOW_LEVEL_FAILURE,
  FC2_ERROR_NOT_FOUND,
  FC2_ERROR_FAILED_GUID,
  FC2_ERROR_INVALID_PACKET_SIZE,
  FC2_ERROR_INVALID_MODE,
  FC2_ERROR_NOT_IN_FORMAT7,
  FC2_ERROR_NOT_SUPPORTED,
  FC2_ERROR_TIMEOUT,
  FC2_ERROR_BUS_MASTER_FAILED,
  FC2_ERROR_INVALID_GENERATION,
  FC2_ERROR_LUT_FAILED,
  FC2_ERROR_IIDC_FAILED,
  FC2_ERROR_STROBE_FAILED,
  FC2_ERROR_TRIGGER_FAILED,
  FC2_ERROR_PROPERTY_FAILED,
  FC2_ERROR_PROPERTY_NOT_PRESENT,
  FC2_ERROR_REGISTER_FAILED,
  FC2_ERROR_READ_REGISTER_FAILED,
  FC2_ERROR_WRITE_REGISTER_FAILED,
  FC2_ERROR_ISOCH_FAILED,
  FC2_ERROR_ISOCH_ALREADY_STARTED,
  FC2_ERROR_ISOCH_NOT_STARTED,
  FC2_ERROR_ISOCH_START_FAILED,
  FC2_ERROR_ISOCH_RETRIEVE_BUFFER_FAILED,
  FC2_ERROR_ISOCH_STOP_FAILED,
  FC2_ERROR_ISOCH_SYNC_FAILED,
  FC2_ERROR_ISOCH_BANDWIDTH_EXCEEDED,
  FC2_ERROR_IMAGE_CONVERSION_FAILED,
  FC2_ERROR_IMAGE_LIBRARY_FAILURE,
  FC2_ERROR_BUFFER_TOO_SMALL,
  FC2_ERROR_IMAGE_CONSISTENCY_ERROR,
  FC2_ERROR_INCOMPATIBLE_DRIVER,
  FC2_ERROR_FORCE_32BITS = FULL_32BIT_VALUE }

  The error types returned by functions.
- enum fc2BusCallbackType {
  FC2_BUS_RESET,
  FC2_ARRIVAL,
  FC2_REMOVAL,
  FC2_CALLBACK_TYPE_FORCE_32BITS = FULL_32BIT_VALUE }

  The type of bus callback to register a callback function for.

- enum fc2GrabMode {
  FC2_DROP_FRAMES,
  FC2_BUFFER_FRAMES,
  FC2_UNSPECIFIED_GRAB_MODE,
  FC2_GRAB_MODE_FORCE_32BITS = FULL_32BIT_VALUE }

  The grab strategy employed during image transfer.
- enum fc2GrabTimeout {
  FC2_TIMEOUT_NONE = 0,
  FC2_TIMEOUT_INFINITE = -1,
  FC2_TIMEOUT_UNSPECIFIED = -2,
  FC2_GRAB_TIMEOUT_FORCE_32BITS = FULL_32BIT_VALUE }

  Timeout options for grabbing images.
- enum fc2BandwidthAllocation {
  FC2_BANDWIDTH_ALLOCATION_OFF = 0,
  FC2_BANDWIDTH_ALLOCATION_ON = 1,
  FC2_BANDWIDTH_ALLOCATION_UNSUPPORTED = 2,
  FC2_BANDWIDTH_ALLOCATION_UNSPECIFIED = 3,
  FC2_BANDWIDTH_ALLOCATION_FORCE_32BITS = FULL_32BIT_VALUE }

  Bandwidth allocation options for 1394 devices.
- enum fc2InterfaceType {
  FC2_INTERFACE_IEEE1394,
  FC2_INTERFACE_USB_2,
  FC2_INTERFACE_USB_3,
  FC2_INTERFACE_GIGE,
  FC2_INTERFACE_UNKNOWN,
  FC2_INTERFACE_TYPE_FORCE_32BITS = FULL_32BIT_VALUE }

  Interfaces that a camera may use to communicate with a host.
- enum fc2PropertyType {
  FC2_BRIGHTNESS,
  FC2_AUTO_EXPOSURE,
  FC2_SHARPNESS,
  FC2_WHITE_BALANCE,
  FC2_HUE,
  FC2_SATURATION,
  FC2_GAMMA,
  FC2_IRIS,
  FC2_FOCUS,
  FC2_ZOOM,
  FC2_PAN,
  FC2_TILT,
  FC2_SHUTTER,
  FC2_GAIN,
  FC2_TRIGGER_MODE,
  FC2_TRIGGER_DELAY,
  FC2_FRAME_RATE,
  FC2_TEMPERATURE,
  FC2_UNSPECIFIED_PROPERTY_TYPE,
  FC2_PROPERTY_TYPE_FORCE_32BITS = FULL_32BIT_VALUE }

  Camera properties.
- enum fc2FrameRate {
  FC2_FRAMERATE_1_875,
  FC2_FRAMERATE_3_75,
  FC2_FRAMERATE_7_5,
  FC2_FRAMERATE_15,
  FC2_FRAMERATE_30,
  FC2_FRAMERATE_60,
  FC2_FRAMERATE_120,

FC2_FRAMERATE_240,
FC2_FRAMERATE_FORMAT7,
FC2_NUM_FRAMERATES,
FC2_FRAMERATE_FORCE_32BITS = FULL_32BIT_VALUE }

Frame rates in frames per second.

- enum fc2VideoMode {
FC2_VIDEOMODE_160x120YUV444,
FC2_VIDEOMODE_320x240YUV422,
FC2_VIDEOMODE_640x480YUV411,
FC2_VIDEOMODE_640x480YUV422,
FC2_VIDEOMODE_640x480RGB,
FC2_VIDEOMODE_640x480Y8,
FC2_VIDEOMODE_640x480Y16,
FC2_VIDEOMODE_800x600YUV422,
FC2_VIDEOMODE_800x600RGB,
FC2_VIDEOMODE_800x600Y8,
FC2_VIDEOMODE_800x600Y16,
FC2_VIDEOMODE_1024x768YUV422,
FC2_VIDEOMODE_1024x768RGB,
FC2_VIDEOMODE_1024x768Y8,
FC2_VIDEOMODE_1024x768Y16,
FC2_VIDEOMODE_1280x960YUV422,
FC2_VIDEOMODE_1280x960RGB,
FC2_VIDEOMODE_1280x960Y8,
FC2_VIDEOMODE_1280x960Y16,
FC2_VIDEOMODE_1600x1200YUV422,
FC2_VIDEOMODE_1600x1200RGB,
FC2_VIDEOMODE_1600x1200Y8,
FC2_VIDEOMODE_1600x1200Y16,
FC2_VIDEOMODE_FORMAT7,
FC2_NUM_VIDEOMODES,
FC2_VIDEOMODE_FORCE_32BITS = FULL_32BIT_VALUE }

DCAM video modes.

- enum fc2Mode {
FC2_MODE_0 = 0,
FC2_MODE_1,
FC2_MODE_2,
FC2_MODE_3,
FC2_MODE_4,
FC2_MODE_5,
FC2_MODE_6,
FC2_MODE_7,
FC2_MODE_8,
FC2_MODE_9,
FC2_MODE_10,
FC2_MODE_11,
FC2_MODE_12,
FC2_MODE_13,
FC2_MODE_14,
FC2_MODE_15,
FC2_MODE_16,
FC2_MODE_17,
FC2_MODE_18,
FC2_MODE_19,
FC2_MODE_20,
FC2_MODE_21,
FC2_MODE_22,
FC2_MODE_23,

FC2_MODE_24,
FC2_MODE_25,
FC2_MODE_26,
FC2_MODE_27,
FC2_MODE_28,
FC2_MODE_29,
FC2_MODE_30,
FC2_MODE_31,
FC2_NUM_MODES,
FC2_MODE_FORCE_32BITS = FULL_32BIT_VALUE }

Camera modes for DCAM formats as well as Format7.

- enum fc2PixelFormat {
FC2_PIXEL_FORMAT_MONO8 = 0x80000000,
FC2_PIXEL_FORMAT_411YUV8 = 0x40000000,
FC2_PIXEL_FORMAT_422YUV8 = 0x20000000,
FC2_PIXEL_FORMAT_444YUV8 = 0x10000000,
FC2_PIXEL_FORMAT_RGB8 = 0x08000000,
FC2_PIXEL_FORMAT_MONO16 = 0x04000000,
FC2_PIXEL_FORMAT_RGB16 = 0x02000000,
FC2_PIXEL_FORMAT_S_MONO16 = 0x01000000,
FC2_PIXEL_FORMAT_S_RGB16 = 0x00800000,
FC2_PIXEL_FORMAT_RAW8 = 0x00400000,
FC2_PIXEL_FORMAT_RAW16 = 0x00200000,
FC2_PIXEL_FORMAT_MONO12 = 0x00100000,
FC2_PIXEL_FORMAT_RAW12 = 0x00080000,
FC2_PIXEL_FORMAT_BGR = 0x80000008,
FC2_PIXEL_FORMAT_BGRU = 0x40000008,
FC2_PIXEL_FORMAT_RGB = FC2_PIXEL_FORMAT_RGB8,
FC2_PIXEL_FORMAT_RGBU = 0x40000002,
FC2_PIXEL_FORMAT_BGR16 = 0x02000001,
FC2_PIXEL_FORMAT_BGRU16 = 0x02000002,
FC2_PIXEL_FORMAT_422YUV8_JPEG = 0x40000001,
FC2_NUM_PIXEL_FORMATS = 20,
FC2_UNSPECIFIED_PIXEL_FORMAT = 0 }

Pixel formats available for Format7 modes.

- enum fc2BusSpeed {
FC2_BUSSPEED_S100,
FC2_BUSSPEED_S200,
FC2_BUSSPEED_S400,
FC2_BUSSPEED_S480,
FC2_BUSSPEED_S800,
FC2_BUSSPEED_S1600,
FC2_BUSSPEED_S3200,
FC2_BUSSPEED_S5000,
FC2_BUSSPEED_10BASE_T,
FC2_BUSSPEED_100BASE_T,
FC2_BUSSPEED_1000BASE_T,
FC2_BUSSPEED_10000BASE_T,
FC2_BUSSPEED_S_FASTEST,
FC2_BUSSPEED_ANY,
FC2_BUSSPEED_SPEED_UNKNOWN = -1,
FC2_BUSSPEED_FORCE_32BITS = FULL_32BIT_VALUE }

Bus speeds.

- enum fc2PCIeBusSpeed {
FC2_PCIE_BUSSPEED_2_5,
FC2_PCIE_BUSSPEED_5_0,
FC2_PCIE_BUSSPEED_UNKNOWN = -1,
FC2_PCIE_BUSSPEED_FORCE_32BITS = FULL_32BIT_VALUE }

- enum fc2DriverType {
  FC2_DRIVER_1394_CAM,
  FC2_DRIVER_1394_PRO,
  FC2_DRIVER_1394_JUJU,
  FC2_DRIVER_1394_VIDEO1394,
  FC2_DRIVER_1394_RAW1394,
  FC2_DRIVER_USB_NONE,
  FC2_DRIVER_USB_CAM,
  FC2_DRIVER_USB3_PRO,
  FC2_DRIVER_GIGE_NONE,
  FC2_DRIVER_GIGE_FILTER,
  FC2_DRIVER_GIGE_PRO,
  FC2_DRIVER_GIGE_LWF,
  FC2_DRIVER_UNKNOWN = -1,
  FC2_DRIVER_FORCE_32BITS = FULL_32BIT_VALUE }

  Types of low level drivers that FlyCapture uses.
- enum fc2ColorProcessingAlgorithm {
  FC2_DEFAULT,
  FC2_NO_COLOR_PROCESSING,
  FC2_NEAREST_NEIGHBOR_FAST,
  FC2_EDGE_SENSING,
  FC2_HQ_LINEAR,
  FC2_RIGOROUS,
  FC2_IPP,
  FC2_DIRECTIONAL,
  FC2_WEIGHTED_DIRECTIONAL,
  FC2_COLOR_PROCESSING_ALGORITHM_FORCE_32BITS = FULL_32BIT_VALUE }

  Color processing algorithms.
- enum fc2BayerTileFormat {
  FC2_BT_NONE,
  FC2_BT_RGGB,
  FC2_BT_GRBG,
  FC2_BT_GBRG,
  FC2_BT_BGGR,
  FC2_BT_FORCE_32BITS = FULL_32BIT_VALUE }

  Bayer tile formats.
- enum fc2ImageFileFormat {
  FC2_FROM_FILE_EXT = -1,
  FC2_PGM,
  FC2_PPM,
  FC2_BMP,
  FC2_JPEG,
  FC2_JPEG2000,
  FC2_TIFF,
  FC2_PNG,
  FC2_RAW,
  FC2_IMAGE_FILE_FORMAT_FORCE_32BITS = FULL_32BIT_VALUE }

  File formats to be used for saving images to disk.

6.22.1 Detailed Description

6.22.2 Enumeration Type Documentation

6.22.2.1 fc2BandwidthAllocation

```
enum fc2BandwidthAllocation
```

Bandwidth allocation options for 1394 devices.

Enumerator

FC2_BANDWIDTH_ALLOCATION_OFF	Do not allocate bandwidth.
FC2_BANDWIDTH_ALLOCATION_ON	Allocate bandwidth. This is the default setting.
FC2_BANDWIDTH_ALLOCATION_UNSUPPORTED	Bandwidth allocation is not supported by either the camera or operating system.
FC2_BANDWIDTH_ALLOCATION_UNSPECIFIED	Not specified. This leaves the current setting unchanged.
FC2_BANDWIDTH_ALLOCATION_FORCE_32BITS	

6.22.2.2 fc2BayerTileFormat

```
enum fc2BayerTileFormat
```

Bayer tile formats.

Enumerator

FC2_BT_NONE	No bayer tile format.
FC2_BT_RGGB	Red-Green-Green-Blue.
FC2_BT_GRBG	Green-Red-Blue-Green.
FC2_BT_GBRG	Green-Blue-Red-Green.
FC2_BT_BGGR	Blue-Green-Green-Red.
FC2_BT_FORCE_32BITS	

6.22.2.3 fc2BusCallbackType

```
enum fc2BusCallbackType
```

The type of bus callback to register a callback function for.

Enumerator

FC2_BUS_RESET	Register for all bus events.
FC2_ARRIVAL	Register for arrivals only.
FC2_REMOVAL	Register for removals only.
FC2_CALLBACK_TYPE_FORCE_32BITS	

6.22.2.4 fc2BusSpeed

```
enum fc2BusSpeed
```

Bus speeds.

Enumerator

FC2_BUSSPEED_S100	100Mbits/sec.
FC2_BUSSPEED_S200	200Mbits/sec.
FC2_BUSSPEED_S400	400Mbits/sec.
FC2_BUSSPEED_S480	480Mbits/sec. Only for USB2 cameras.
FC2_BUSSPEED_S800	800Mbits/sec.
FC2_BUSSPEED_S1600	1600Mbits/sec.
FC2_BUSSPEED_S3200	3200Mbits/sec.
FC2_BUSSPEED_S5000	5000Mbits/sec. Only for USB3 cameras.
FC2_BUSSPEED_10BASE_T	10Base-T. Only for GigE cameras.
FC2_BUSSPEED_100BASE_T	100Base-T. Only for GigE cameras.
FC2_BUSSPEED_1000BASE_T	1000Base-T (Gigabit Ethernet). Only for GigE cameras.
FC2_BUSSPEED_10000BASE_T	10000Base-T. Only for GigE cameras.
FC2_BUSSPEED_S_FASTEST	The fastest speed available.
FC2_BUSSPEED_ANY	Any speed that is available.
FC2_BUSSPEED_SPEED_UNKNOWN	Unknown bus speed.
FC2_BUSSPEED_FORCE_32BITS	

6.22.2.5 fc2ColorProcessingAlgorithm

```
enum fc2ColorProcessingAlgorithm
```

Color processing algorithms.

Please refer to our knowledge base at article at http://www.ptgrey.com/support/kb/index.↩
asp?a=4&q=33 for complete details for each algorithm.

Enumerator

FC2_DEFAULT	Default method.
FC2_NO_COLOR_PROCESSING	No color processing.
FC2_NEAREST_NEIGHBOR_FAST	Fastest but lowest quality. Equivalent to FLYCAPTURE_NEAREST_NEIGHBOR_FAST in FlyCapture.
FC2_EDGE_SENSING	Weights surrounding pixels based on localized edge orientation.
FC2_HQ_LINEAR	Well-balanced speed and quality.
FC2_RIGOROUS	Slowest but produces good results.

Enumerator

FC2_IPP	Multithreaded with similar results to edge sensing.
FC2_DIRECTIONAL	Best quality but much faster than rigorous.
FC2_WEIGHTED_DIRECTIONAL	Weighted pixel average from different directions.
FC2_COLOR_PROCESSING_ALGORITHM_FOR↩ CE_32BITS	

6.22.2.6 fc2DriverType

```
enum fc2DriverType
```

Types of low level drivers that FlyCapture uses.

Enumerator

FC2_DRIVER_1394_CAM	PGRCam.sys.
FC2_DRIVER_1394_PRO	PGR1394.sys.
FC2_DRIVER_1394_JUJU	firewire_core.
FC2_DRIVER_1394_VIDEO1394	video1394.
FC2_DRIVER_1394_RAW1394	raw1394.
FC2_DRIVER_USB_NONE	No usb driver used just BSD stack. (Linux only)
FC2_DRIVER_USB_CAM	PGRUsbCam.sys.
FC2_DRIVER_USB3_PRO	PGRXHCI.sys.
FC2_DRIVER_GIGE_NONE	no GigE drivers used, MS/BSD stack.
FC2_DRIVER_GIGE_FILTER	PGRGigE.sys.
FC2_DRIVER_GIGE_PRO	PGRGigEPro.sys.
FC2_DRIVER_GIGE_LWF	PgrLwf.sys.
FC2_DRIVER_UNKNOWN	Unknown driver type.
FC2_DRIVER_FORCE_32BITS	

6.22.2.7 fc2Error

```
enum fc2Error
```

The error types returned by functions.

Enumerator

FC2_ERROR_UNDEFINED	Undefined.
FC2_ERROR_OK	Function returned with no errors.
FC2_ERROR_FAILED	General failure.
FC2_ERROR_NOT_IMPLEMENTED	Function has not been implemented.
FC2_ERROR_FAILED_BUS_MASTER_CONNECTION	Could not connect to Bus Master.

Enumerator

FC2_ERROR_NOT_CONNECTED	Camera has not been connected.
FC2_ERROR_INIT_FAILED	Initialization failed.
FC2_ERROR_NOT_INTITIALIZED	Camera has not been initialized.
FC2_ERROR_INVALID_PARAMETER	Invalid parameter passed to function.
FC2_ERROR_INVALID_SETTINGS	Setting set to camera is invalid.
FC2_ERROR_INVALID_BUS_MANAGER	Invalid Bus Manager object.
FC2_ERROR_MEMORY_ALLOCATION_FAILED	Could not allocate memory.
FC2_ERROR_LOW_LEVEL_FAILURE	Low level error.
FC2_ERROR_NOT_FOUND	Device not found.
FC2_ERROR_FAILED_GUID	GUID failure.
FC2_ERROR_INVALID_PACKET_SIZE	Packet size set to camera is invalid.
FC2_ERROR_INVALID_MODE	Invalid mode has been passed to function.
FC2_ERROR_NOT_IN_FORMAT7	Error due to not being in Format7.
FC2_ERROR_NOT_SUPPORTED	This feature is unsupported.
FC2_ERROR_TIMEOUT	Timeout error.
FC2_ERROR_BUS_MASTER_FAILED	Bus Master Failure.
FC2_ERROR_INVALID_GENERATION	Generation Count Mismatch.
FC2_ERROR_LUT_FAILED	Look Up Table failure.
FC2_ERROR_IIDC_FAILED	IIDC failure.
FC2_ERROR_STROBE_FAILED	Strobe failure.
FC2_ERROR_TRIGGER_FAILED	Trigger failure.
FC2_ERROR_PROPERTY_FAILED	Property failure.
FC2_ERROR_PROPERTY_NOT_PRESENT	Property is not present.
FC2_ERROR_REGISTER_FAILED	Register access failed.
FC2_ERROR_READ_REGISTER_FAILED	Register read failed.
FC2_ERROR_WRITE_REGISTER_FAILED	Register write failed.
FC2_ERROR_ISOCH_FAILED	Isochronous failure.
FC2_ERROR_ISOCH_ALREADY_STARTED	Isochronous transfer has already been started.
FC2_ERROR_ISOCH_NOT_STARTED	Isochronous transfer has not been started.
FC2_ERROR_ISOCH_START_FAILED	Isochronous start failed.
FC2_ERROR_ISOCH_RETRIEVE_BUFFER_FAILED	Isochronous retrieve buffer failed.
FC2_ERROR_ISOCH_STOP_FAILED	Isochronous stop failed.
FC2_ERROR_ISOCH_SYNC_FAILED	Isochronous image synchronization failed.
FC2_ERROR_ISOCH_BANDWIDTH_EXCEEDED	Isochronous bandwidth exceeded.
FC2_ERROR_IMAGE_CONVERSION_FAILED	Image conversion failed.
FC2_ERROR_IMAGE_LIBRARY_FAILURE	Image library failure.
FC2_ERROR_BUFFER_TOO_SMALL	Buffer is too small.
FC2_ERROR_IMAGE_CONSISTENCY_ERROR	There is an image consistency error.
FC2_ERROR_INCOMPATIBLE_DRIVER	The installed driver is not compatible with the library.
FC2_ERROR_FORCE_32BITS	

6.22.2.8 fc2FrameRate

```
enum fc2FrameRate
```

Frame rates in frames per second.

Enumerator

FC2_FRAMERATE_1_875	1.875 fps.
FC2_FRAMERATE_3_75	3.75 fps.
FC2_FRAMERATE_7_5	7.5 fps.
FC2_FRAMERATE_15	15 fps.
FC2_FRAMERATE_30	30 fps.
FC2_FRAMERATE_60	60 fps.
FC2_FRAMERATE_120	120 fps.
FC2_FRAMERATE_240	240 fps.
FC2_FRAMERATE_FORMAT7	Custom frame rate for Format7 functionality.
FC2_NUM_FRAMERATES	Number of possible camera frame rates.
FC2_FRAMERATE_FORCE_32BITS	

6.22.2.9 fc2GrabMode

```
enum fc2GrabMode
```

The grab strategy employed during image transfer.

This type controls how images that stream off the camera accumulate in a user buffer for handling.

Enumerator

FC2_DROP_FRAMES	Grabs the newest image in the user buffer each time the RetrieveBuffer() function is called. Older images are dropped instead of accumulating in the user buffer. Grabbing blocks if the camera has not finished transmitting the next available image. If the camera is transmitting images faster than the application can grab them, images may be dropped and only the most recent image is stored for grabbing. Note that this mode is the equivalent of flycaptureLockLatest in earlier versions of the FlyCapture SDK.
FC2_BUFFER_FRAMES	Images accumulate in the user buffer, and the oldest image is grabbed for handling before being discarded. This member can be used to guarantee that each image is seen. However, image processing time must not exceed transmission time from the camera to the buffer. Grabbing blocks if the camera has not finished transmitting the next available image. The buffer size is controlled by the numBuffers parameter in the FC2Config struct. Note that this mode is the equivalent of flycaptureLockNext in earlier versions of the FlyCapture SDK.
FC2_UNSPECIFIED_GRAB_MODE	Unspecified grab mode.
FC2_GRAB_MODE_FORCE_32BITS	

6.22.2.10 fc2GrabTimeout

```
enum fc2GrabTimeout
```

Timeout options for grabbing images.

Enumerator

FC2_TIMEOUT_NONE	Non-blocking wait.
FC2_TIMEOUT_INFINITE	Wait indefinitely.
FC2_TIMEOUT_UNSPECIFIED	Unspecified timeout setting.
FC2_GRAB_TIMEOUT_FORCE_32BITS	

6.22.2.11 fc2ImageFileFormat

```
enum fc2ImageFileFormat
```

File formats to be used for saving images to disk.

Enumerator

FC2_FROM_FILE_EXT	Determine file format from file extension.
FC2_PGM	Portable gray map.
FC2_PPM	Portable pixmap.
FC2_BMP	Bitmap.
FC2_JPEG	JPEG.
FC2_JPEG2000	JPEG 2000.
FC2_TIFF	Tagged image file format.
FC2_PNG	Portable network graphics.
FC2_RAW	Raw data.
FC2_IMAGE_FILE_FORMAT_FORCE_32BITS	

6.22.2.12 fc2InterfaceType

```
enum fc2InterfaceType
```

Interfaces that a camera may use to communicate with a host.

Enumerator

FC2_INTERFACE_IEEE1394	IEEE-1394 (Includes 1394a and 1394b).
FC2_INTERFACE_USB_2	USB 2.0.
FC2_INTERFACE_USB_3	USB 3.0.
FC2_INTERFACE_GIGE	GigE.
FC2_INTERFACE_UNKNOWN	Unknown interface.
FC2_INTERFACE_TYPE_FORCE_32BITS	

6.22.2.13 fc2Mode

```
enum fc2Mode
```

Camera modes for DCAM formats as well as Format7.

Enumerator

FC2_MODE_0	
FC2_MODE_1	
FC2_MODE_2	
FC2_MODE_3	
FC2_MODE_4	
FC2_MODE_5	
FC2_MODE_6	
FC2_MODE_7	
FC2_MODE_8	
FC2_MODE_9	
FC2_MODE_10	
FC2_MODE_11	
FC2_MODE_12	
FC2_MODE_13	
FC2_MODE_14	
FC2_MODE_15	
FC2_MODE_16	
FC2_MODE_17	
FC2_MODE_18	
FC2_MODE_19	
FC2_MODE_20	
FC2_MODE_21	
FC2_MODE_22	
FC2_MODE_23	
FC2_MODE_24	
FC2_MODE_25	
FC2_MODE_26	
FC2_MODE_27	
FC2_MODE_28	
FC2_MODE_29	
FC2_MODE_30	
FC2_MODE_31	
FC2_NUM_MODES	Number of modes.
FC2_MODE_FORCE_32BITS	

6.22.2.14 fc2PCIeBusSpeed

```
enum fc2PCIeBusSpeed
```

Enumerator

FC2_PCIE_BUSSPEED_2_5	
FC2_PCIE_BUSSPEED_5_0	2.5 Gb/s
FC2_PCIE_BUSSPEED_UNKNOWN	5.0 Gb/s
FC2_PCIE_BUSSPEED_FORCE_32BITS	Speed is unknown.

6.22.2.15 fc2PixelFormat

`enum fc2PixelFormat`

Pixel formats available for Format7 modes.

Enumerator

FC2_PIXEL_FORMAT_MONO8	8 bits of mono information.
FC2_PIXEL_FORMAT_411YUV8	YUV 4:1:1.
FC2_PIXEL_FORMAT_422YUV8	YUV 4:2:2.
FC2_PIXEL_FORMAT_444YUV8	YUV 4:4:4.
FC2_PIXEL_FORMAT_RGB8	R = G = B = 8 bits.
FC2_PIXEL_FORMAT_MONO16	16 bits of mono information.
FC2_PIXEL_FORMAT_RGB16	R = G = B = 16 bits.
FC2_PIXEL_FORMAT_S_MONO16	16 bits of signed mono information.
FC2_PIXEL_FORMAT_S_RGB16	R = G = B = 16 bits signed.
FC2_PIXEL_FORMAT_RAW8	8 bit raw data output of sensor.
FC2_PIXEL_FORMAT_RAW16	16 bit raw data output of sensor.
FC2_PIXEL_FORMAT_MONO12	12 bits of mono information.
FC2_PIXEL_FORMAT_RAW12	12 bit raw data output of sensor.
FC2_PIXEL_FORMAT_BGR	24 bit BGR.
FC2_PIXEL_FORMAT_BGRU	32 bit BGRU.
FC2_PIXEL_FORMAT_RGB	24 bit RGB.
FC2_PIXEL_FORMAT_RGBU	32 bit RGBU.
FC2_PIXEL_FORMAT_BGR16	R = G = B = 16 bits.
FC2_PIXEL_FORMAT_BGRU16	64 bit BGRU.
FC2_PIXEL_FORMAT_422YUV8_JPEG	JPEG compressed stream.
FC2_NUM_PIXEL_FORMATS	Number of pixel formats.
FC2_UNSPECIFIED_PIXEL_FORMAT	Unspecified pixel format.

6.22.2.16 fc2PropertyType

`enum fc2PropertyType`

Camera properties.

Not all properties may be supported, depending on the camera model.

Enumerator

FC2_BRIGHTNESS	
FC2_AUTO_EXPOSURE	
FC2_SHARPNESS	
FC2_WHITE_BALANCE	
FC2_HUE	
FC2_SATURATION	
FC2_GAMMA	
FC2_IRIS	
FC2_FOCUS	
FC2_ZOOM	
FC2_PAN	
FC2_TILT	
FC2_SHUTTER	
FC2_GAIN	
FC2_TRIGGER_MODE	
FC2_TRIGGER_DELAY	
FC2_FRAME_RATE	
FC2_TEMPERATURE	
FC2_UNSPECIFIED_PROPERTY_TYPE	
FC2_PROPERTY_TYPE_FORCE_32BITS	

6.22.2.17 fc2VideoMode

```
enum fc2VideoMode
```

DCAM video modes.

Enumerator

FC2_VIDEOMODE_160x120YUV444	160x120 YUV444.
FC2_VIDEOMODE_320x240YUV422	320x240 YUV422.
FC2_VIDEOMODE_640x480YUV411	640x480 YUV411.
FC2_VIDEOMODE_640x480YUV422	640x480 YUV422.
FC2_VIDEOMODE_640x480RGB	640x480 24-bit RGB.
FC2_VIDEOMODE_640x480Y8	640x480 8-bit.
FC2_VIDEOMODE_640x480Y16	640x480 16-bit.
FC2_VIDEOMODE_800x600YUV422	800x600 YUV422.
FC2_VIDEOMODE_800x600RGB	800x600 RGB.
FC2_VIDEOMODE_800x600Y8	800x600 8-bit.
FC2_VIDEOMODE_800x600Y16	800x600 16-bit.
FC2_VIDEOMODE_1024x768YUV422	1024x768 YUV422.
FC2_VIDEOMODE_1024x768RGB	1024x768 RGB.
FC2_VIDEOMODE_1024x768Y8	1024x768 8-bit.
FC2_VIDEOMODE_1024x768Y16	1024x768 16-bit.
FC2_VIDEOMODE_1280x960YUV422	1280x960 YUV422.
FC2_VIDEOMODE_1280x960RGB	1280x960 RGB.

Enumerator

FC2_VIDEOMODE_1280x960Y8	1280x960 8-bit.
FC2_VIDEOMODE_1280x960Y16	1280x960 16-bit.
FC2_VIDEOMODE_1600x1200YUV422	1600x1200 YUV422.
FC2_VIDEOMODE_1600x1200RGB	1600x1200 RGB.
FC2_VIDEOMODE_1600x1200Y8	1600x1200 8-bit.
FC2_VIDEOMODE_1600x1200Y16	1600x1200 16-bit.
FC2_VIDEOMODE_FORMAT7	Custom video mode for Format7 functionality.
FC2_NUM_VIDEOMODES	Number of possible video modes.
FC2_VIDEOMODE_FORCE_32BITS	

6.23 GigE specific enumerations

These enumerations are specific to GigE camera operation only.

Enumerations

- enum fc2GigEPropertyType {
  FC2_HEARTBEAT,
  FC2_HEARTBEAT_TIMEOUT,
  PACKET_SIZE,
  PACKET_DELAY }

  Possible properties that can be queried from the camera.

6.23.1 Detailed Description

These enumerations are specific to GigE camera operation only.

6.23.2 Enumeration Type Documentation

6.23.2.1 fc2GigEPropertyType

```
enum fc2GigEPropertyType
```

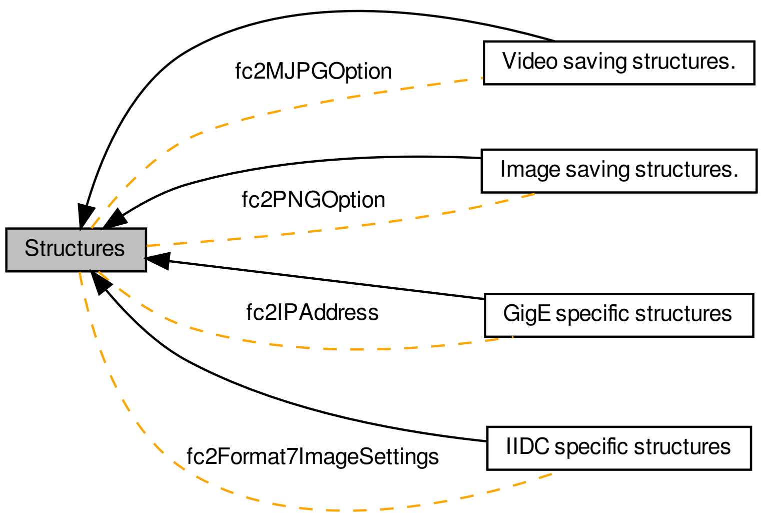
Possible properties that can be queried from the camera.

Enumerator

FC2_HEARTBEAT	
FC2_HEARTBEAT_TIMEOUT	
PACKET_SIZE	
PACKET_DELAY	

6.24 Structures

Collaboration diagram for Structures:



Modules

- GigE specific structures

  These structures are specific to GigE camera operation only.
- IIDC specific structures

  These structures are specific to IIDC camera operation only.
- Image saving structures.

  These structures define various parameters used for saving images.
- Video saving structures.

  These structures define various parameters used for saving videos.

Data Structures

- struct fc2Image
- struct fc2SystemInfo

  Description of the system.
- struct fc2Version

  The current version of the library.
- struct fc2IPAddress

  IPv4 address.
- struct fc2Format7ImageSettings

  Format 7 image settings.
- struct fc2Config

  Configuration for a camera.
- struct fc2TriggerDelayInfo

Information about a specific camera property.
- struct fc2TriggerDelay

  A specific camera property.
- struct fc2TriggerModeInfo

  Information about a camera trigger property.
- struct fc2TriggerMode

  A camera trigger.
- struct fc2StrobeInfo

  A camera strobe property.
- struct fc2StrobeControl

  A camera strobe.
- struct fc2TimeStamp

  Timestamp information.
- struct fc2ConfigROM

  Camera configuration ROM.
- struct fc2CameraInfo

  Camera information.
- struct fc2EmbeddedImageInfoProperty

  Properties of a single embedded image info property.
- struct fc2EmbeddedImageInfo

  Properties of the possible embedded image information.
- struct fc2ImageMetadata

  Metadata related to an image.
- struct fc2LUTData

  Information about the camera's look up table.
- struct fc2CameraStats

  Camera diagnostic information.
- struct fc2PNGOption

  Options for saving PNG images.
- struct fc2MJPGOption

  Options for saving MJPG files.

6.24.1 Detailed Description

6.25 GigE specific structures

These structures are specific to GigE camera operation only.

Collaboration diagram for GigE specific structures:

Structures
fc2IPAddress
GigE specific structures

Data Structures

- struct fc2IPAddress

  IPv4 address.
- struct fc2MACAddress

  MAC address.
- struct fc2GigEProperty

  A GigE property.
- struct fc2GigEStreamChannel

  Information about a single GigE stream channel.
- struct fc2GigEConfig

  Configuration for a GigE camera.
- struct fc2GigEImageSettingsInfo

  Format 7 information for a single mode.
- struct fc2GigEImageSettings

  Image settings for a GigE camera.

6.25.1 Detailed Description

These structures are specific to GigE camera operation only.

6.26 IIDC specific structures

These structures are specific to IIDC camera operation only.

Collaboration diagram for IIDC specific structures:

Structures
fc2Format7ImageSettings
IIDC specific structures

Data Structures

- struct fc2Format7ImageSettings

  Format 7 image settings.
- struct fc2Format7Info

  Format 7 information for a single mode.
- struct fc2Format7PacketInfo

  Format 7 packet information.

6.26.1 Detailed Description

These structures are specific to IIDC camera operation only.

6.27 Image saving structures.

These structures define various parameters used for saving images.

Collaboration diagram for Image saving structures.:

Data Structures

- struct fc2PNGOption

  Options for saving PNG images.
- struct fc2PPMOption

  Options for saving PPM images.
- struct fc2PGMOption

  Options for saving PGM images.
- struct fc2TIFFOption

  Options for saving TIFF images.
- struct fc2JPEGOption

  Options for saving JPEG image.
- struct fc2JPG2Option

  Options for saving JPEG2000 image.
- struct fc2BMPOption

  Options for saving Bitmap image.
- struct fc2EventOptions

  Options for enabling device event registration.
- struct fc2EventCallbackData

Typedefs

- typedef void ∗ fc2CallbackHandle
- typedef void(∗ fc2BusEventCallback) (void ∗pParameter, unsigned int serialNumber)
- typedef void(∗ fc2ImageEventCallback) (fc2Image ∗image, void ∗pCallbackData)
- typedef void(∗ fc2AsyncCommandCallback) (fc2Error retError, void ∗pUserData)
- typedef void(∗ fc2CameraEventCallback) (void ∗pCallbackData)

Enumerations

- enum fc2TIFFCompressionMethod {
  FC2_TIFF_NONE = 1,
  FC2_TIFF_PACKBITS,
  FC2_TIFF_DEFLATE,
  FC2_TIFF_ADOBE_DEFLATE,
  FC2_TIFF_CCITTFAX3,
  FC2_TIFF_CCITTFAX4,
  FC2_TIFF_LZW,
  FC2_TIFF_JPEG }

6.27.1 Detailed Description

These structures define various parameters used for saving images.

6.27.2 Typedef Documentation

6.27.2.1 fc2AsyncCommandCallback

```
typedef void(* fc2AsyncCommandCallback) (fc2Error retError, void *pUserData)
```

6.27.2.2 fc2BusEventCallback

```
typedef void(* fc2BusEventCallback) (void *pParameter, unsigned int serialNumber)
```

6.27.2.3 fc2CallbackHandle

```
typedef void* fc2CallbackHandle
```

6.27.2.4 fc2CameraEventCallback

```
typedef void(* fc2CameraEventCallback) (void *pCallbackData)
```

6.27.2.5 fc2ImageEventCallback

```
typedef void(* fc2ImageEventCallback) (fc2Image *image, void *pCallbackData)
```

6.27.3 Enumeration Type Documentation

6.27.3.1 fc2TIFFCompressionMethod

```
enum fc2TIFFCompressionMethod
```

Enumerator

FC2_TIFF_NONE	Save without any compression.
FC2_TIFF_PACKBITS	Save using PACKBITS compression.
FC2_TIFF_DEFLATE	Save using DEFLATE compression (ZLIB compression).
FC2_TIFF_ADOBE_DEFLATE	Save using ADOBE DEFLATE compression.
FC2_TIFF_CCITTFAX3	Save using CCITT Group 3 fax encoding. This is only valid for 1-bit images only. Default to LZW for other bit depths.
FC2_TIFF_CCITTFAX4	Save using CCITT Group 4 fax encoding. This is only valid for 1-bit images only. Default to LZW for other bit depths.
FC2_TIFF_LZW	Save using LZW compression.
FC2_TIFF_JPEG	Save using JPEG compression. This is only valid for 8-bit greyscale and 24-bit only. Default to LZW for other bit depths.

6.28 Video Recording Operation

The video recording operation provides the functionality for the user to record images to an video file.

Functions

- FLYCAPTURE2_C_API fc2Error fc2VideoCreate (fc2VideoContext ∗pVideoContext)

    Create a Video context.
- FLYCAPTURE2_C_API fc2Error fc2VideoAVIOpen (fc2VideoContext VideoContext, const char ∗pFileName, fc2AVIOption ∗pOption)

    Open an AVI file in preparation for writing Images to disk.
- FLYCAPTURE2_C_API fc2Error fc2VideoMJPGOpen (fc2VideoContext VideoContext, const char ∗pFile←Name, fc2MJPGOption ∗pOption)

    Open an MJPEG file in preparation for writing Images to disk.
- FLYCAPTURE2_C_API fc2Error fc2VideoH264Open (fc2VideoContext VideoContext, const char ∗pFile←Name, fc2H264Option ∗pOption)

    Open an H.264 video file in preparation for writing Images to disk.
- FLYCAPTURE2_C_API fc2Error fc2VideoAppend (fc2VideoContext VideoContext, fc2Image ∗pImage)

    Append an image to the video file.
- FLYCAPTURE2_C_API fc2Error fc2VideoSetMaximumSize (fc2VideoContext VideoContext, unsigned int size)

    Set the maximum file size (in megabytes) of a AVI/MP4 file.
- FLYCAPTURE2_C_API fc2Error fc2VideoClose (fc2VideoContext VideoContext)

    Close the video file.
- FLYCAPTURE2_C_API fc2Error fc2VideoDestroy (fc2VideoContext VideoContext)

    Destroy a video context.

6.28.1 Detailed Description

The video recording operation provides the functionality for the user to record images to an video file.

6.28.2 Function Documentation

6.28.2.1 fc2VideoAppend()

```
FLYCAPTURE2_C_API fc2Error fc2VideoAppend (
            fc2VideoContext VideoContext,
            fc2Image * pImage )
```

Append an image to the video file.

Parameters

VideoContext	The video context to use.
pImage	The image to append.

Returns

A fc2Error indicating the success or failure of the function.

6.28.2.2 fc2VideoAVIOpen()

```
FLYCAPTURE2_C_API fc2Error fc2VideoAVIOpen (
             fc2VideoContext VideoContext,
             const char * pFileName,
             fc2AVIOption * pOption )
```

Open an AVI file in preparation for writing Images to disk.

The size of AVI files is limited to 2GB. The filenames are automatically generated using the filename specified.

Parameters

VideoContext	The video context to use.
pFileName	The filename of the AVI file.
pOption	Options to apply to the AVI file.

See also

SetMaximumFileSize()
fc2Close()
fc2AVIOption

Returns

A fc2Error indicating the success or failure of the function.

6.28.2.3 fc2VideoClose()

```
FLYCAPTURE2_C_API fc2Error fc2VideoClose (
             fc2VideoContext VideoContext )
```

Close the video file.

Parameters

VideoContext	The video context to use.

Returns

A fc2Error indicating the success or failure of the function.

6.28.2.4 fc2VideoCreate()

```
FLYCAPTURE2_C_API fc2Error fc2VideoCreate (
             fc2VideoContext * pVideoContext )
```

Create a Video context.

Parameters

pVideoContext	A video context.

Returns

A fc2Error indicating the success or failure of the function.

6.28.2.5 fc2VideoDestroy()

```
FLYCAPTURE2_C_API fc2Error fc2VideoDestroy (
             fc2VideoContext VideoContext )
```

Destroy a video context.

Parameters

VideoContext	A video context.

Returns

A fc2Error indicating the success or failure of the function.

6.28.2.6 fc2VideoH264Open()

```
FLYCAPTURE2_C_API fc2Error fc2VideoH264Open (
             fc2VideoContext VideoContext,
             const char * pFileName,
             fc2H264Option * pOption )
```

Open an H.264 video file in preparation for writing Images to disk.

If the file extension is not specified, MP4 will be used as the default container. Consult ffmpeg documentation for a list of supported containers.

Parameters

pFileName	The filename of the video file.
pOption	H.264 options to apply to the video file.

See also

fc2Close()
fc2H264Option

Returns

A fc2Error indicating the success or failure of the function.

6.28.2.7 fc2VideoMJPGOpen()

```
FLYCAPTURE2_C_API fc2Error fc2VideoMJPGOpen (
            fc2VideoContext VideoContext,
            const char * pFileName,
            fc2MJPGOption * pOption )
```

Open an MJPEG file in preparation for writing Images to disk.

The size of AVI files is limited to 2GB. The filenames are automatically generated using the filename specified.

Parameters

VideoContext	The AVI context to use.
pFileName	The filename of the AVI file.
pOption	Options to apply to the AVI file.

See also

fc2Close()
fc2MJPGOption

Returns

A fc2Error indicating the success or failure of the function.

6.28.2.8 fc2VideoSetMaximumSize()

```
FLYCAPTURE2_C_API fc2Error fc2VideoSetMaximumSize (
            fc2VideoContext VideoContext,
            unsigned int size )
```

Set the maximum file size (in megabytes) of a AVI/MP4 file.

A new AVI/MP4 file is created automatically when file size limit is reached. Setting a maximum size of 0 indicates no limit on file size.

Parameters

VideoContext	The video context to use.
size	The maximum video file size in MB.

Returns

A fc2Error indicating the success or failure of the function.

6.29 Video saving structures.

These structures define various parameters used for saving videos.

Collaboration diagram for Video saving structures.:

Data Structures

- struct fc2MJPGOption

  Options for saving MJPG files.
- struct fc2H264Option

  Options for saving H264 files.
- struct fc2AVIOption

  Options for saving AVI files.

6.29.1 Detailed Description

These structures define various parameters used for saving videos.

Chapter 7

Data Structure Documentation

7.1 fc2AVIOption Struct Reference

Options for saving AVI files.

Data Fields

- float frameRate

  Frame rate of the stream.
- unsigned int reserved [256]

  Reserved for future use.

7.1.1 Detailed Description

Options for saving AVI files.

7.1.2 Field Documentation

7.1.2.1 frameRate

```
float frameRate
```

Frame rate of the stream.

7.1.2.2 reserved

```
unsigned int reserved[256]
```

Reserved for future use.

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

- FlyCapture2VideoDefs_C.h

7.2 fc2BMPOption Struct Reference

Options for saving Bitmap image.

Data Fields

- BOOL indexedColor_8bit
- unsigned int reserved [16]

  Reserved for future use.

7.2.1 Detailed Description

Options for saving Bitmap image.

7.2.2 Field Documentation

7.2.2.1 indexedColor_8bit

```
BOOL indexedColor_8bit
```

7.2.2.2 reserved

```
unsigned int reserved[16]
```

Reserved for future use.

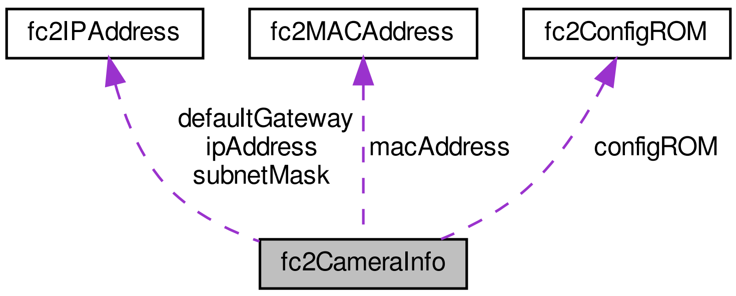
The documentation for this struct was generated from the following file:

- FlyCapture2Defs_C.h

7.3 fc2CameraInfo Struct Reference

Camera information.

Collaboration diagram for fc2CameraInfo:



Data Fields

- unsigned int serialNumber

  Device serial number.
- fc2InterfaceType interfaceType

  Interface type.
- fc2DriverType driverType

  Driver type.
- BOOL isColorCamera

  Flag indicating if this is a color camera.
- char modelName [MAX_STRING_LENGTH]

  Device model name.
- char vendorName [MAX_STRING_LENGTH]

  Device vendor name.
- char sensorInfo [MAX_STRING_LENGTH]

  String detailing the sensor information.
- char sensorResolution [MAX_STRING_LENGTH]

  String providing the sensor resolution.
- char driverName [MAX_STRING_LENGTH]

  Driver name of driver being used.
- char firmwareVersion [MAX_STRING_LENGTH]

  Firmware version of camera.
- char firmwareBuildTime [MAX_STRING_LENGTH]

  Firmware build time.
- fc2BusSpeed maximumBusSpeed

  Maximum bus speed.
- fc2BayerTileFormat bayerTileFormat

  Bayer tile format.
- fc2PCIeBusSpeed pcieBusSpeed

Bus number, set to 0 for GigE and USB cameras.

- unsigned short nodeNumber

  ieee1394 Node number, set to 0 for GigE and USB cameras

- unsigned short busNumber

  PCIe Bus Speed, set to PCIE_BUSSPEED_UNKNOWN for unsupported drivers.

- unsigned int reserved [16]

  Reserved for future use.

IIDC specific information

- unsigned int iidcVer

  DCAM version.
- fc2ConfigROM configROM

  Configuration ROM data.

GigE specific information

- unsigned int gigEMajorVersion

  GigE Vision version.
- unsigned int gigEMinorVersion

  GigE Vision minor version.
- char userDefinedName [MAX_STRING_LENGTH]

  User defined name.
- char xmlURL1 [MAX_STRING_LENGTH]

  XML URL 1.
- char xmlURL2 [MAX_STRING_LENGTH]

  XML URL 2.
- fc2MACAddress macAddress

  MAC address.
- fc2IPAddress ipAddress

  IP address.
- fc2IPAddress subnetMask

  Subnet mask.
- fc2IPAddress defaultGateway

  Default gateway.
- unsigned int ccpStatus

  Status/Content of CCP register.
- unsigned int applicationIPAddress

  Local Application IP Address.
- unsigned int applicationPort

  Local Application port.

7.3.1 Detailed Description

Camera information.

7.3.2 Field Documentation

7.3.2.1 applicationIPAddress

```
unsigned int applicationIPAddress
```

Local Application IP Address.

7.3.2.2 applicationPort

```
unsigned int applicationPort
```

Local Application port.

7.3.2.3 bayerTileFormat

```
fc2BayerTileFormat bayerTileFormat
```

Bayer tile format.

7.3.2.4 busNumber

```
unsigned short busNumber
```

PCIe Bus Speed, set to PCIE_BUSSPEED_UNKNOWN for unsupported drivers.

7.3.2.5 ccpStatus

```
unsigned int ccpStatus
```

Status/Content of CCP register.

7.3.2.6 configROM

```
fc2ConfigROM configROM
```

Configuration ROM data.

7.3.2.7 defaultGateway

```
fc2IPAddress defaultGateway
```

Default gateway.

7.3.2.8 driverName

```
char driverName[MAX_STRING_LENGTH]
```

Driver name of driver being used.

7.3.2.9 driverType

```
fc2DriverType driverType
```

Driver type.

7.3.2.10 firmwareBuildTime

```
char firmwareBuildTime[MAX_STRING_LENGTH]
```

Firmware build time.

7.3.2.11 firmwareVersion

```
char firmwareVersion[MAX_STRING_LENGTH]
```

Firmware version of camera.

7.3.2.12 gigEMajorVersion

```
unsigned int gigEMajorVersion
```

GigE Vision version.

7.3.2.13 gigEMinorVersion

```
unsigned int gigEMinorVersion
```

GigE Vision minor version.

7.3.2.14 iidcVer

```
unsigned int iidcVer
```

DCAM version.

7.3.2.15 interfaceType

```
fc2InterfaceType interfaceType
```

Interface type.

7.3.2.16 ipAddress

```
fc2IPAddress ipAddress
```

IP address.

7.3.2.17 isColorCamera

```
BOOL isColorCamera
```

Flag indicating if this is a color camera.

7.3.2.18 macAddress

```
fc2MACAddress macAddress
```

MAC address.

7.3.2.19 maximumBusSpeed

```
fc2BusSpeed maximumBusSpeed
```

Maximum bus speed.

7.3.2.20 modelName

```
char modelName[MAX_STRING_LENGTH]
```

Device model name.

7.3.2.21 nodeNumber

```
unsigned short nodeNumber
```

ieee1394 Node number, set to 0 for GigE and USB cameras

7.3.2.22 pcieBusSpeed

```
fc2PCIeBusSpeed pcieBusSpeed
```

Bus number, set to 0 for GigE and USB cameras.

7.3.2.23 reserved

```
unsigned int reserved[16]
```

Reserved for future use.

7.3.2.24 sensorInfo

```
char sensorInfo[MAX_STRING_LENGTH]
```

String detailing the sensor information.

7.3.2.25 sensorResolution

```
char sensorResolution[MAX_STRING_LENGTH]
```

String providing the sensor resolution.

7.3.2.26 serialNumber

```
unsigned int serialNumber
```

Device serial number.

7.3.2.27 subnetMask

```
fc2IPAddress subnetMask
```

Subnet mask.

7.3.2.28 userDefinedName

```
char userDefinedName[MAX_STRING_LENGTH]
```

User defined name.

7.3.2.29 vendorName

```
char vendorName[MAX_STRING_LENGTH]
```

Device vendor name.

7.3.2.30 xmlURL1

```
char xmlURL1[MAX_STRING_LENGTH]
```

XML URL 1.

7.3.2.31 xmlURL2

```
char xmlURL2[MAX_STRING_LENGTH]
```

XML URL 2.

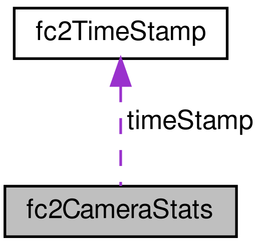
The documentation for this struct was generated from the following file:

- FlyCapture2Defs_C.h

7.4 fc2CameraStats Struct Reference

Camera diagnostic information.

Collaboration diagram for fc2CameraStats:



Data Fields

- unsigned int imageDropped
- unsigned int imageCorrupt
- unsigned int imageXmitFailed
- unsigned int imageDriverDropped
- unsigned int regReadFailed
- unsigned int regWriteFailed
- unsigned int portErrors
- BOOL cameraPowerUp
- float cameraVoltages [8]
- unsigned int numVoltages

  The number of voltage registers available.
- float cameraCurrents [8]
- unsigned int numCurrents

  The number of current registers available.
- unsigned int temperature
- unsigned int timeSinceInitialization
- unsigned int timeSinceBusReset
- fc2TimeStamp timeStamp
- unsigned int numResendPacketsRequested
- unsigned int numResendPacketsReceived
- unsigned int reserved [16]

  Reserved for future use.

7.4.1 Detailed Description

Camera diagnostic information.

7.4.2 Field Documentation

7.4.2.1 cameraCurrents

```
float cameraCurrents[8]
```

7.4.2.2 cameraPowerUp

```
BOOL cameraPowerUp
```

7.4.2.3 cameraVoltages

```
float cameraVoltages[8]
```

7.4.2.4 imageCorrupt

```
unsigned int imageCorrupt
```

7.4.2.5 imageDriverDropped

```
unsigned int imageDriverDropped
```

7.4.2.6 imageDropped

```
unsigned int imageDropped
```

7.4.2.7 imageXmitFailed

```
unsigned int imageXmitFailed
```

7.4.2.8 numCurrents

```
unsigned int numCurrents
```

The number of current registers available.

0: the values in cameraCurrents[] are invalid.

7.4.2.9 numResendPacketsReceived

```
unsigned int numResendPacketsReceived
```

7.4.2.10 numResendPacketsRequested

```
unsigned int numResendPacketsRequested
```

7.4.2.11 numVoltages

```
unsigned int numVoltages
```

The number of voltage registers available.

0: the values in cameraVoltages[] are invalid.

7.4.2.12 portErrors

```
unsigned int portErrors
```

7.4.2.13 regReadFailed

```
unsigned int regReadFailed
```

7.4.2.14 regWriteFailed

```
unsigned int regWriteFailed
```

7.4.2.15 reserved

```
unsigned int reserved[16]
```

Reserved for future use.

7.4.2.16 temperature

```
unsigned int temperature
```

7.4.2.17 timeSinceBusReset

```
unsigned int timeSinceBusReset
```

7.4.2.18 timeSinceInitialization

```
unsigned int timeSinceInitialization
```

7.4.2.19 timeStamp

```
fc2TimeStamp timeStamp
```

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

- FlyCapture2Defs_C.h

7.5 fc2Config Struct Reference

Configuration for a camera.

Data Fields

- unsigned int numBuffers

  Number of buffers used by the FlyCapture2 library to grab images.
- unsigned int numImageNotifications

  Number of notifications per image.
- unsigned int minNumImageNotifications

  Minimum number of notifications needed for the current image settings on the camera.
- int grabTimeout

  Time in milliseconds that RetrieveBuffer() and WaitForBufferEvent() will wait for an image before timing out and returning.
- fc2GrabMode grabMode

  Grab mode for the camera.
- BOOL highPerformanceRetrieveBuffer

  This parameter enables RetrieveBuffer to run in high performance mode.
- fc2BusSpeed isochBusSpeed

  Isochronous bus speed.
- fc2BusSpeed asyncBusSpeed

  Asynchronous bus speed.
- fc2BandwidthAllocation bandwidthAllocation

  Bandwidth allocation flag that tells the camera the bandwidth allocation strategy to employ.
- unsigned int registerTimeoutRetries

  Number of retries to perform when a register read/write timeout is received by the library.
- unsigned int registerTimeout

  Register read/write timeout value, in microseconds.
- unsigned int reserved [16]

  Reserved for future use.

7.5.1 Detailed Description

Configuration for a camera.

These options are options that are generally should be set before starting isochronous transfer.

7.5.2 Field Documentation

7.5.2.1 asyncBusSpeed

```
fc2BusSpeed asyncBusSpeed
```

Asynchronous bus speed.

7.5.2.2 bandwidthAllocation

```
fc2BandwidthAllocation bandwidthAllocation
```

Bandwidth allocation flag that tells the camera the bandwidth allocation strategy to employ.

7.5.2.3 grabMode

```
fc2GrabMode grabMode
```

Grab mode for the camera.

The default is DROP_FRAMES.

7.5.2.4 grabTimeout

```
int grabTimeout
```

Time in milliseconds that RetrieveBuffer() and WaitForBufferEvent() will wait for an image before timing out and returning.

7.5.2.5 highPerformanceRetrieveBuffer

```
BOOL highPerformanceRetrieveBuffer
```

This parameter enables RetrieveBuffer to run in high performance mode.

This means that any interaction with the camera, other than grabbing the image is disabled. Currently Retrieve buffer reads registers on the camera to determine which embedded image information settings have been enabled, and it reads what the bayer tile is currently set to. When High Performance mode is on, these reads are disabled. This means that any changes to the Bayer Tile or to the Embedded image info after StartCapture() will not be tracked when made using direct register writes. If the corresponding SetEmbededImageInfo() and GetEmbededImageInfo() calls are used then the changes will be appropriately reflected. This also means that changes to embedded image info from other processes will not be updated either.

7.5.2.6 isochBusSpeed

```
fc2BusSpeed isochBusSpeed
```

Isochronous bus speed.

7.5.2.7 minNumImageNotifications

```
unsigned int minNumImageNotifications
```

Minimum number of notifications needed for the current image settings on the camera.

Read-only value.

7.5.2.8 numBuffers

```
unsigned int numBuffers
```

Number of buffers used by the FlyCapture2 library to grab images.

7.5.2.9 numImageNotifications

```
unsigned int numImageNotifications
```

Number of notifications per image.

This value should only be set after the image settings to be used is set to the camera. The default number of notifications is 1.

There are 4 general scenarios:

- 1 notification - End of image
- 2 notifications - After first packet and end of image
- 3 notifications - After first packet, middle of image, end of image
- x notifications - After first packet, (x -2) spread evenly, end of image

Specifying zero for the number of notifications will be ignored (the current value will not be modified).

Note that the event numbers start at 0. Ex. when 3 notifications are used, the three events will be 0, 1 and 2.

7.5.2.10 registerTimeout

```
unsigned int registerTimeout
```

Register read/write timeout value, in microseconds.

The default value is dependent on the interface type.

7.5.2.11 registerTimeoutRetries

```
unsigned int registerTimeoutRetries
```

Number of retries to perform when a register read/write timeout is received by the library.

The default value is 0.

7.5.2.12 reserved

```
unsigned int reserved[16]
```

Reserved for future use.

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

- FlyCapture2Defs_C.h

7.6 fc2ConfigROM Struct Reference

Camera configuration ROM.

Data Fields

- unsigned int nodeVendorId

  Vendor ID of a node.
- unsigned int chipIdHi

  Chip ID (high part).
- unsigned int chipIdLo

  Chip ID (low part).
- unsigned int unitSpecId

  Unit Spec ID, usually 0xa02d.
- unsigned int unitSWVer

  Unit software version.
- unsigned int unitSubSWVer

  Unit sub software version.
- unsigned int vendorUniqueInfo_0

  Vendor unique info 0.
- unsigned int vendorUniqueInfo_1

  Vendor unique info 1.
- unsigned int vendorUniqueInfo_2

  Vendor unique info 2.
- unsigned int vendorUniqueInfo_3

  Vendor unique info 3.
- char pszKeyword [MAX_STRING_LENGTH]

  Keyword.
- unsigned int reserved [16]

  Reserved for future use.

7.6.1 Detailed Description

Camera configuration ROM.

7.6.2 Field Documentation

7.6.2.1 chipIdHi

```
unsigned int chipIdHi
```

Chip ID (high part).

7.6.2.2 chipIdLo

```
unsigned int chipIdLo
```

Chip ID (low part).

7.6.2.3 nodeVendorId

```
unsigned int nodeVendorId
```

Vendor ID of a node.

7.6.2.4 pszKeyword

```
char pszKeyword[MAX_STRING_LENGTH]
```

Keyword.

7.6.2.5 reserved

```
unsigned int reserved[16]
```

Reserved for future use.

7.6.2.6 unitSpecId

```
unsigned int unitSpecId
```

Unit Spec ID, usually 0xa02d.

7.6.2.7 unitSubSWVer

```
unsigned int unitSubSWVer
```

Unit sub software version.

7.6.2.8 unitSWVer

```
unsigned int unitSWVer
```

Unit software version.

7.6.2.9 vendorUniqueInfo_0

```
unsigned int vendorUniqueInfo_0
```

Vendor unique info 0.

7.6.2.10 vendorUniqueInfo_1

```
unsigned int vendorUniqueInfo_1
```

Vendor unique info 1.

7.6.2.11 vendorUniqueInfo_2

```
unsigned int vendorUniqueInfo_2
```

Vendor unique info 2.

7.6.2.12 vendorUniqueInfo_3

```
unsigned int vendorUniqueInfo_3
```

Vendor unique info 3.

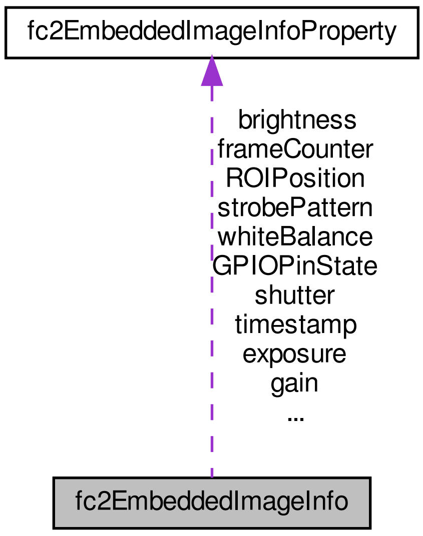
The documentation for this struct was generated from the following file:

- FlyCapture2Defs_C.h

7.7 fc2EmbeddedImageInfo Struct Reference

Properties of the possible embedded image information.

Collaboration diagram for fc2EmbeddedImageInfo:



Data Fields

- fc2EmbeddedImageInfoProperty timestamp
- fc2EmbeddedImageInfoProperty gain
- fc2EmbeddedImageInfoProperty shutter
- fc2EmbeddedImageInfoProperty brightness
- fc2EmbeddedImageInfoProperty exposure
- fc2EmbeddedImageInfoProperty whiteBalance
- fc2EmbeddedImageInfoProperty frameCounter
- fc2EmbeddedImageInfoProperty strobePattern
- fc2EmbeddedImageInfoProperty GPIOPinState
- fc2EmbeddedImageInfoProperty ROIPosition

7.7.1 Detailed Description

Properties of the possible embedded image information.

7.7.2 Field Documentation

7.7.2.1 brightness

`fc2EmbeddedImageInfoProperty brightness`

7.7.2.2 exposure

`fc2EmbeddedImageInfoProperty exposure`

7.7.2.3 frameCounter

`fc2EmbeddedImageInfoProperty frameCounter`

7.7.2.4 gain

`fc2EmbeddedImageInfoProperty gain`

7.7.2.5 GPIOPinState

`fc2EmbeddedImageInfoProperty GPIOPinState`

7.7.2.6 ROIPosition

`fc2EmbeddedImageInfoProperty ROIPosition`

7.7.2.7 shutter

```
fc2EmbeddedImageInfoProperty shutter
```

7.7.2.8 strobePattern

```
fc2EmbeddedImageInfoProperty strobePattern
```

7.7.2.9 timestamp

```
fc2EmbeddedImageInfoProperty timestamp
```

7.7.2.10 whiteBalance

```
fc2EmbeddedImageInfoProperty whiteBalance
```

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

- FlyCapture2Defs_C.h

7.8 fc2EmbeddedImageInfoProperty Struct Reference

Properties of a single embedded image info property.

Data Fields

- BOOL available

  Whether this property is available.
- BOOL onOff

  Whether this property is on or off.

7.8.1 Detailed Description

Properties of a single embedded image info property.

7.8.2 Field Documentation

7.8.2.1 available

```
BOOL available
```

Whether this property is available.

7.8.2.2 onOff

```
BOOL onOff
```

Whether this property is on or off.

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

- FlyCapture2Defs_C.h

7.9 fc2EventCallbackData Struct Reference

Data Fields

- void ∗ EventUserData

  Pointer to the user-supplied data struct.
- size_t EventUserDataSize

  Size of the user data supplied to the RegisterEvent() function.
- const char ∗ EventName

  The event name used to register the event.
- unsigned long long EventID

  The device register which EventName maps to.
- unsigned long long EventTimestamp

  Timestamp indicated the time (as reported by the camera) at which the camera exposure operation completed.
- void ∗ EventData

  A pointer to additional data pertaining to the event which just trigger the callback function.
- size_t EventDataSize

  The size of the structure pointed to by EventData.

7.9.1 Field Documentation

7.9.1.1 EventData

```
void∗ EventData
```

A pointer to additional data pertaining to the event which just trigger the callback function.

The data may be of difference sizes or may not even be allocated, depending on the type of event which triggered the callback.

7.9.1.2 EventDataSize

```
size_t EventDataSize
```

The size of the structure pointed to by EventData.

This value should be checked, especially if there are events which can trigger variable- length event data to be returned to the user when the callback function is issued.

7.9.1.3 EventID

```
unsigned long long EventID
```

The device register which EventName maps to.

Provides an alternate means of indexing into different event types.

7.9.1.4 EventName

```
const char* EventName
```

The event name used to register the event.

Provided so the user knows which event triggered the callback.

7.9.1.5 EventTimestamp

```
unsigned long long EventTimestamp
```

Timestamp indicated the time (as reported by the camera) at which the camera exposure operation completed.

This can be compared with image timestamps if there is a need to map event timestamps to specific images, if applicable.

7.9.1.6 EventUserData

```
void* EventUserData
```

Pointer to the user-supplied data struct.

7.9.1.7 EventUserDataSize

```
size_t EventUserDataSize
```

Size of the user data supplied to the RegisterEvent() function.

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

- FlyCapture2Defs_C.h

7.10 fc2EventOptions Struct Reference

Options for enabling device event registration.

Data Fields

- fc2CameraEventCallback EventCallbackFcn

  Callback function pointer.
- const char ∗ EventName

  Event name to register.
- const void ∗ EventUserData

  Pointer to callback data to be passed to the callback function.
- size_t EventUserDataSize

  Size of the underlying struct passed as eventCallbackData for sanity checks.

7.10.1 Detailed Description

Options for enabling device event registration.

7.10.2 Field Documentation

7.10.2.1 EventCallbackFcn

```
fc2CameraEventCallback EventCallbackFcn
```

Callback function pointer.

7.10.2.2 EventName

```
const char* EventName
```

Event name to register.

7.10.2.3 EventUserData

```
const void* EventUserData
```

Pointer to callback data to be passed to the callback function.

7.10.2.4 EventUserDataSize

```
size_t EventUserDataSize
```

Size of the underlying struct passed as eventCallbackData for sanity checks.

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

- FlyCapture2Defs_C.h

7.11 fc2Format7ImageSettings Struct Reference

Format 7 image settings.

Data Fields

- fc2Mode mode

  Format 7 mode.
- unsigned int offsetX

  Horizontal image offset.
- unsigned int offsetY

  Vertical image offset.
- unsigned int width

  Width of image.
- unsigned int height

  Height of image.
- fc2PixelFormat pixelFormat

  Pixel format of image.
- unsigned int reserved [8]

  Reserved for future use.

7.11.1 Detailed Description

Format 7 image settings.

7.11.2 Field Documentation

7.11.2.1 height

```
unsigned int height
```

Height of image.

7.11.2.2 mode

```
fc2Mode mode
```

Format 7 mode.

7.11.2.3 offsetX

```
unsigned int offsetX
```

Horizontal image offset.

7.11.2.4 offsetY

```
unsigned int offsetY
```

Vertical image offset.

7.11.2.5 pixelFormat

```
fc2PixelFormat pixelFormat
```

Pixel format of image.

7.11.2.6 reserved

```
unsigned int reserved[8]
```

Reserved for future use.

7.11.2.7 width

```
unsigned int width
```

Width of image.

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

- FlyCapture2Defs_C.h

7.12 fc2Format7Info Struct Reference

Format 7 information for a single mode.

Data Fields

- fc2Mode mode

  Format 7 mode.
- unsigned int maxWidth

  Maximum image width.
- unsigned int maxHeight

  Maximum image height.
- unsigned int offsetHStepSize

  Horizontal step size for the offset.
- unsigned int offsetVStepSize

  Vertical step size for the offset.
- unsigned int imageHStepSize

  Horizontal step size for the image.
- unsigned int imageVStepSize

  Vertical step size for the image.
- unsigned int pixelFormatBitField

  Supported pixel formats in a bit field.
- unsigned int vendorPixelFormatBitField

  Vendor unique pixel formats in a bit field.
- unsigned int packetSize

  Current packet size in bytes.
- unsigned int minPacketSize

  Minimum packet size in bytes for current mode.
- unsigned int maxPacketSize

  Maximum packet size in bytes for current mode.
- float percentage

  Current packet size as a percentage of maximum packet size.
- unsigned int reserved [16]

  Reserved for future use.

7.12.1 Detailed Description

Format 7 information for a single mode.

7.12.2 Field Documentation

7.12.2.1 imageHStepSize

```
unsigned int imageHStepSize
```

Horizontal step size for the image.

7.12.2.2 imageVStepSize

```
unsigned int imageVStepSize
```

Vertical step size for the image.

7.12.2.3 maxHeight

```
unsigned int maxHeight
```

Maximum image height.

7.12.2.4 maxPacketSize

```
unsigned int maxPacketSize
```

Maximum packet size in bytes for current mode.

7.12.2.5 maxWidth

```
unsigned int maxWidth
```

Maximum image width.

7.12.2.6 minPacketSize

```
unsigned int minPacketSize
```

Minimum packet size in bytes for current mode.

7.12.2.7 mode

```
fc2Mode mode
```

Format 7 mode.

7.12.2.8 offsetHStepSize

```
unsigned int offsetHStepSize
```

Horizontal step size for the offset.

7.12.2.9 offsetVStepSize

```
unsigned int offsetVStepSize
```

Vertical step size for the offset.

7.12.2.10 packetSize

```
unsigned int packetSize
```

Current packet size in bytes.

7.12.2.11 percentage

```
float percentage
```

Current packet size as a percentage of maximum packet size.

7.12.2.12 pixelFormatBitField

```
unsigned int pixelFormatBitField
```

Supported pixel formats in a bit field.

7.12.2.13 reserved

```
unsigned int reserved[16]
```

Reserved for future use.

7.12.2.14 vendorPixelFormatBitField

```
unsigned int vendorPixelFormatBitField
```

Vendor unique pixel formats in a bit field.

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

- FlyCapture2Defs_C.h

7.13 fc2Format7PacketInfo Struct Reference

Format 7 packet information.

Data Fields

- unsigned int recommendedBytesPerPacket

  Recommended bytes per packet.
- unsigned int maxBytesPerPacket

  Maximum bytes per packet.
- unsigned int unitBytesPerPacket

  Minimum bytes per packet.
- unsigned int reserved [8]

  Reserved for future use.

7.13.1 Detailed Description

Format 7 packet information.

7.13.2 Field Documentation

7.13.2.1 maxBytesPerPacket

```
unsigned int maxBytesPerPacket
```

Maximum bytes per packet.

7.13.2.2 recommendedBytesPerPacket

```
unsigned int recommendedBytesPerPacket
```

Recommended bytes per packet.

7.13.2.3 reserved

```
unsigned int reserved[8]
```

Reserved for future use.

7.13.2.4 unitBytesPerPacket

```
unsigned int unitBytesPerPacket
```

Minimum bytes per packet.

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

- FlyCapture2Defs_C.h

7.14 fc2GigEConfig Struct Reference

Configuration for a GigE camera.

Data Fields

- BOOL enablePacketResend
  Turn on/off packet resend functionality.
- unsigned int registerTimeoutRetries
  Number of retries to perform when a register read/write timeout is received by the library.
- unsigned int registerTimeout
  Register read/write timeout value, in microseconds.
- unsigned int reserved [8]

7.14.1 Detailed Description

Configuration for a GigE camera.

These options are options that are generally should be set before starting isochronous transfer.

7.14.2 Field Documentation

7.14.2.1 enablePacketResend

```
BOOL enablePacketResend
```

Turn on/off packet resend functionality.

7.14.2.2 registerTimeout

```
unsigned int registerTimeout
```

Register read/write timeout value, in microseconds.

The default value is dependent on the interface type.

7.14.2.3 registerTimeoutRetries

```
unsigned int registerTimeoutRetries
```

Number of retries to perform when a register read/write timeout is received by the library.

The default value is 0.

7.14.2.4 reserved

```
unsigned int reserved[8]
```

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

- FlyCapture2Defs_C.h

7.15 fc2GigEImageSettings Struct Reference

Image settings for a GigE camera.

Data Fields

- unsigned int offsetX

  Horizontal image offset.
- unsigned int offsetY

  Vertical image offset.
- unsigned int width

  Width of image.
- unsigned int height

  Height of image.
- fc2PixelFormat pixelFormat

  Pixel format of image.
- unsigned int reserved [8]

  Reserved for future use.

7.15.1 Detailed Description

Image settings for a GigE camera.

7.15.2 Field Documentation

7.15.2.1 height

```
unsigned int height
```

Height of image.

7.15.2.2 offsetX

```
unsigned int offsetX
```

Horizontal image offset.

7.15.2.3 offsetY

```
unsigned int offsetY
```

Vertical image offset.

7.15.2.4 pixelFormat

`fc2PixelFormat pixelFormat`

Pixel format of image.

7.15.2.5 reserved

`unsigned int reserved[8]`

Reserved for future use.

7.15.2.6 width

`unsigned int width`

Width of image.

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

- FlyCapture2Defs_C.h

7.16 fc2GigEImageSettingsInfo Struct Reference

Format 7 information for a single mode.

Data Fields

- unsigned int maxWidth

  Maximum image width.
- unsigned int maxHeight

  Maximum image height.
- unsigned int offsetHStepSize

  Horizontal step size for the offset.
- unsigned int offsetVStepSize

  Vertical step size for the offset.
- unsigned int imageHStepSize

  Horizontal step size for the image.
- unsigned int imageVStepSize

  Vertical step size for the image.
- unsigned int pixelFormatBitField

  Supported pixel formats in a bit field.
- unsigned int vendorPixelFormatBitField

  Vendor unique pixel formats in a bit field.
- unsigned int reserved [16]

  Reserved for future use.

7.16.1 Detailed Description

Format 7 information for a single mode.

7.16.2 Field Documentation

7.16.2.1 imageHStepSize

```
unsigned int imageHStepSize
```

Horizontal step size for the image.

7.16.2.2 imageVStepSize

```
unsigned int imageVStepSize
```

Vertical step size for the image.

7.16.2.3 maxHeight

```
unsigned int maxHeight
```

Maximum image height.

7.16.2.4 maxWidth

```
unsigned int maxWidth
```

Maximum image width.

7.16.2.5 offsetHStepSize

```
unsigned int offsetHStepSize
```

Horizontal step size for the offset.

7.16.2.6 offsetVStepSize

```
unsigned int offsetVStepSize
```

Vertical step size for the offset.

7.16.2.7 pixelFormatBitField

```
unsigned int pixelFormatBitField
```

Supported pixel formats in a bit field.

7.16.2.8 reserved

```
unsigned int reserved[16]
```

Reserved for future use.

7.16.2.9 vendorPixelFormatBitField

```
unsigned int vendorPixelFormatBitField
```

Vendor unique pixel formats in a bit field.

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

- FlyCapture2Defs_C.h

7.17 fc2GigEProperty Struct Reference

A GigE property.

Data Fields

- fc2GigEPropertyType propType

  The type of property.
- BOOL isReadable

  Whether the property is readable.
- BOOL isWritable

  Whether the property is writable.
- unsigned int min

  Minimum value.
- unsigned int max

  Maximum value.
- unsigned int value

  Current value.
- unsigned int reserved [8]

7.17.1 Detailed Description

A GigE property.

7.17.2 Field Documentation

7.17.2.1 isReadable

```
BOOL isReadable
```

Whether the property is readable.

If this is false, then no other value in this structure is valid.

7.17.2.2 isWritable

```
BOOL isWritable
```

Whether the property is writable.

7.17.2.3 max

```
unsigned int max
```

Maximum value.

7.17.2.4 min

```
unsigned int min
```

Minimum value.

7.17.2.5 propType

```
fc2GigEPropertyType propType
```

The type of property.

7.17.2.6 reserved

```
unsigned int reserved[8]
```

7.17.2.7 value

```
unsigned int value
```

Current value.

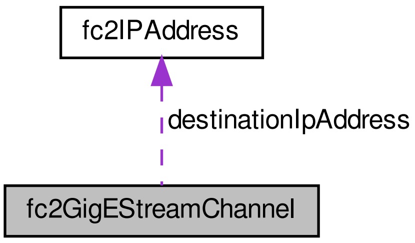
The documentation for this struct was generated from the following file:

- FlyCapture2Defs_C.h

7.18 fc2GigEStreamChannel Struct Reference

Information about a single GigE stream channel.

Collaboration diagram for fc2GigEStreamChannel:



Data Fields

- unsigned int networkInterfaceIndex

  Network interface index used (or to use).
- unsigned int hostPort

  Host port on the PC where the camera will send the data stream.
- BOOL doNotFragment

  Disable IP fragmentation of packets.
- unsigned int packetSize

  Packet size, in bytes.
- unsigned int interPacketDelay

  Inter packet delay, in timestamp counter units.
- fc2IPAddress destinationIpAddress

  Destination IP address.
- unsigned int sourcePort

  Source UDP port of the stream channel.
- unsigned int reserved [8]

7.18.1 Detailed Description

Information about a single GigE stream channel.

7.18.2 Field Documentation

7.18.2.1 destinationIpAddress

```
fc2IPAddress destinationIpAddress
```

Destination IP address.

It can be a multicast or unicast address.

7.18.2.2 doNotFragment

```
BOOL doNotFragment
```

Disable IP fragmentation of packets.

7.18.2.3 hostPort

```
unsigned int hostPort
```

Host port on the PC where the camera will send the data stream.

7.18.2.4 interPacketDelay

```
unsigned int interPacketDelay
```

Inter packet delay, in timestamp counter units.

7.18.2.5 networkInterfaceIndex

```
unsigned int networkInterfaceIndex
```

Network interface index used (or to use).

7.18.2.6 packetSize

```
unsigned int packetSize
```

Packet size, in bytes.

7.18.2.7 reserved

```
unsigned int reserved[8]
```

7.18.2.8 sourcePort

```
unsigned int sourcePort
```

Source UDP port of the stream channel.

Read only.

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

- FlyCapture2Defs_C.h

7.19 fc2H264Option Struct Reference

Options for saving H264 files.

Data Fields

- float frameRate

  Frame rate of the stream.
- unsigned int width

  Width of source image.
- unsigned int height

  Height of source image.
- unsigned int bitrate

  Bitrate to encode at.
- unsigned int reserved [256]

  Reserved for future use.

7.19.1 Detailed Description

Options for saving H264 files.

7.19.2 Field Documentation

7.19.2.1 bitrate

```
unsigned int bitrate
```

Bitrate to encode at.

7.19.2.2 frameRate

```
float frameRate
```

Frame rate of the stream.

7.19.2.3 height

```
unsigned int height
```

Height of source image.

7.19.2.4 reserved

```
unsigned int reserved[256]
```

Reserved for future use.

7.19.2.5 width

```
unsigned int width
```

Width of source image.

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

- FlyCapture2VideoDefs_C.h

7.20 fc2Image Struct Reference

Data Fields

- unsigned int rows
- unsigned int cols
- unsigned int stride
- unsigned char ∗ pData
- unsigned int dataSize
- unsigned int receivedDataSize
- fc2PixelFormat format
- fc2BayerTileFormat bayerFormat
- fc2ImageImpl imageImpl

7.20.1 Field Documentation

7.20.1.1 bayerFormat

```
fc2BayerTileFormat bayerFormat
```

7.20.1.2 cols

```
unsigned int cols
```

7.20.1.3 dataSize

```
unsigned int dataSize
```

7.20.1.4 format

```
fc2PixelFormat format
```

7.20.1.5 imageImpl

```
fc2ImageImpl imageImpl
```

7.20.1.6 pData

```
unsigned char* pData
```

7.20.1.7 receivedDataSize

```
unsigned int receivedDataSize
```

7.20.1.8 rows

```
unsigned int rows
```

7.20.1.9 stride

```
unsigned int stride
```

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

- FlyCapture2Defs_C.h

7.21 fc2ImageMetadata Struct Reference

Metadata related to an image.

Data Fields

- unsigned int embeddedTimeStamp

    Embedded timestamp.
- unsigned int embeddedGain

    Embedded gain.
- unsigned int embeddedShutter

    Embedded shutter.
- unsigned int embeddedBrightness

    Embedded brightness.
- unsigned int embeddedExposure

    Embedded exposure.
- unsigned int embeddedWhiteBalance

    Embedded white balance.
- unsigned int embeddedFrameCounter

    Embedded frame counter.
- unsigned int embeddedStrobePattern

    Embedded strobe pattern.
- unsigned int embeddedGPIOPinState

    Embedded GPIO pin state.
- unsigned int embeddedROIPosition

    Embedded ROI position.
- unsigned int reserved [31]

    Reserved for future use.

7.21.1 Detailed Description

Metadata related to an image.

7.21.2 Field Documentation

7.21.2.1 embeddedBrightness

```
unsigned int embeddedBrightness
```

Embedded brightness.

7.21.2.2 embeddedExposure

```
unsigned int embeddedExposure
```

Embedded exposure.

7.21.2.3 embeddedFrameCounter

```
unsigned int embeddedFrameCounter
```

Embedded frame counter.

7.21.2.4 embeddedGain

```
unsigned int embeddedGain
```

Embedded gain.

7.21.2.5 embeddedGPIOPinState

```
unsigned int embeddedGPIOPinState
```

Embedded GPIO pin state.

7.21.2.6 embeddedROIPosition

```
unsigned int embeddedROIPosition
```

Embedded ROI position.

7.21.2.7 embeddedShutter

```
unsigned int embeddedShutter
```

Embedded shutter.

7.21.2.8 embeddedStrobePattern

```
unsigned int embeddedStrobePattern
```

Embedded strobe pattern.

7.21.2.9 embeddedTimeStamp

```
unsigned int embeddedTimeStamp
```

Embedded timestamp.

7.21.2.10 embeddedWhiteBalance

```
unsigned int embeddedWhiteBalance
```

Embedded white balance.

7.21.2.11 reserved

```
unsigned int reserved[31]
```

Reserved for future use.

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

- FlyCapture2Defs_C.h

7.22 fc2InternalContext Struct Reference

Data Fields

- FlyCapture2::BusManager ∗ pBusMgr
- FlyCapture2::CameraBase ∗ pCamera

7.22.1 Field Documentation

7.22.1.1 pBusMgr

```
FlyCapture2::BusManager* pBusMgr
```

7.22.1.2 pCamera

```
FlyCapture2::CameraBase* pCamera
```

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

- FlyCapture2Internal_C.h

7.23 fc2InternalGuiContext Struct Reference

Data Fields

- FlyCapture2::CameraSelectionDlg ∗ pCameraSelectionDlg
- FlyCapture2::CameraControlDlg ∗ pCameraControlDlg

7.23.1 Field Documentation

7.23.1.1 pCameraControlDlg

```
FlyCapture2::CameraControlDlg* pCameraControlDlg
```

7.23.1.2 pCameraSelectionDlg

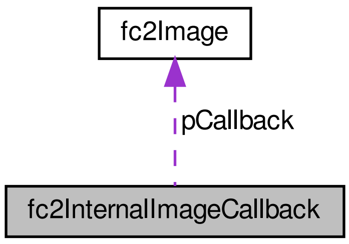
```
FlyCapture2::CameraSelectionDlg* pCameraSelectionDlg
```

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

- FlyCapture2Internal_C.h

7.24 fc2InternalImageCallback Struct Reference

Collaboration diagram for fc2InternalImageCallback:



Data Fields

- fc2ImageEventCallback pCallback
- void ∗ pCallbackData

7.24.1 Field Documentation

7.24.1.1 pCallback

```
fc2ImageEventCallback pCallback
```

7.24.1.2 pCallbackData

```
void* pCallbackData
```

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

- FlyCapture2Internal_C.h

7.25 fc2IPAddress Struct Reference

IPv4 address.

Data Fields

- unsigned char octets [4]

7.25.1 Detailed Description

IPv4 address.

7.25.2 Field Documentation

7.25.2.1 octets

```
unsigned char octets[4]
```

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

- FlyCapture2Defs_C.h

7.26 fc2JPEGOption Struct Reference

Options for saving JPEG image.

Data Fields

- BOOL progressive

  Whether to save as a progressive JPEG file.
- unsigned int quality

  JPEG image quality in range (0-100).
- unsigned int reserved [16]

  Reserved for future use.

7.26.1 Detailed Description

Options for saving JPEG image.

7.26.2 Field Documentation

7.26.2.1 progressive

```
BOOL progressive
```

Whether to save as a progressive JPEG file.

7.26.2.2 quality

```
unsigned int quality
```

JPEG image quality in range (0-100).

- 100 - Superb quality.
- 75 - Good quality.
- 50 - Normal quality.
- 10 - Poor quality.

7.26.2.3 reserved

```
unsigned int reserved[16]
```

Reserved for future use.

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

- FlyCapture2Defs_C.h

7.27 fc2JPG2Option Struct Reference

Options for saving JPEG2000 image.

Data Fields

- unsigned int quality

    JPEG saving quality in range (1-512).
- unsigned int reserved [16]

    Reserved for future use.

7.27.1 Detailed Description

Options for saving JPEG2000 image.

7.27.2 Field Documentation

7.27.2.1 quality

```
unsigned int quality
```

JPEG saving quality in range (1-512).

7.27.2.2 reserved

```
unsigned int reserved[16]
```

Reserved for future use.

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

- FlyCapture2Defs_C.h

7.28 fc2LUTData Struct Reference

Information about the camera's look up table.

Data Fields

- BOOL supported

  Flag indicating if LUT is supported.
- BOOL enabled

  Flag indicating if LUT is enabled.
- unsigned int numBanks

  The number of LUT banks available (Always 1 for PGR LUT).
- unsigned int numChannels

  The number of LUT channels per bank available.
- unsigned int inputBitDepth

  The input bit depth of the LUT.
- unsigned int outputBitDepth

  The output bit depth of the LUT.
- unsigned int numEntries

  The number of entries in the LUT.
- unsigned int reserved [8]

  Reserved for future use.

7.28.1 Detailed Description

Information about the camera's look up table.

7.28.2 Field Documentation

7.28.2.1 enabled

```
BOOL enabled
```

Flag indicating if LUT is enabled.

7.28.2.2 inputBitDepth

```
unsigned int inputBitDepth
```

The input bit depth of the LUT.

7.28.2.3 numBanks

```
unsigned int numBanks
```

The number of LUT banks available (Always 1 for PGR LUT).

7.28.2.4 numChannels

```
unsigned int numChannels
```

The number of LUT channels per bank available.

7.28.2.5 numEntries

```
unsigned int numEntries
```

The number of entries in the LUT.

7.28.2.6 outputBitDepth

```
unsigned int outputBitDepth
```

The output bit depth of the LUT.

7.28.2.7 reserved

```
unsigned int reserved[8]
```

Reserved for future use.

7.28.2.8 supported

```
BOOL supported
```

Flag indicating if LUT is supported.

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

- FlyCapture2Defs_C.h

7.29 fc2MACAddress Struct Reference

MAC address.

Data Fields

- unsigned char octets [6]

7.29.1 Detailed Description

MAC address.

7.29.2 Field Documentation

7.29.2.1 octets

```
unsigned char octets[6]
```

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

- FlyCapture2Defs_C.h

7.30 fc2MJPGOption Struct Reference

Options for saving MJPG files.

Data Fields

- float frameRate

  Frame rate of the stream.
- unsigned int quality

  Image quality (1-100)
- unsigned int reserved [256]

7.30.1 Detailed Description

Options for saving MJPG files.

7.30.2 Field Documentation

7.30.2.1 frameRate

```
float frameRate
```

Frame rate of the stream.

7.30.2.2 quality

```
unsigned int quality
```

Image quality (1-100)

7.30.2.3 reserved

```
unsigned int reserved[256]
```

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

- FlyCapture2VideoDefs_C.h

7.31 fc2PGMOption Struct Reference

Options for saving PGM images.

Data Fields

- BOOL binaryFile

  Whether to save the PPM as a binary file.
- unsigned int reserved [16]

  Reserved for future use.

7.31.1 Detailed Description

Options for saving PGM images.

7.31.2 Field Documentation

7.31.2.1 binaryFile

```
BOOL binaryFile
```

Whether to save the PPM as a binary file.

7.31.2.2 reserved

```
unsigned int reserved[16]
```

Reserved for future use.

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

- FlyCapture2Defs_C.h

7.32 fc2PGRGuid Struct Reference

A GUID to the camera.

Data Fields

- unsigned int value [4]

7.32.1 Detailed Description

A GUID to the camera.

It is used to uniquely identify a camera.

7.32.2 Field Documentation

7.32.2.1 value

```
unsigned int value[4]
```

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

- FlyCapture2Defs_C.h

7.33 fc2PNGOption Struct Reference

Options for saving PNG images.

Data Fields

- BOOL interlaced

  Whether to save the PNG as interlaced.
- unsigned int compressionLevel

  Compression level (0-9).
- unsigned int reserved [16]

  Reserved for future use.

7.33.1 Detailed Description

Options for saving PNG images.

7.33.2 Field Documentation

7.33.2.1 compressionLevel

```
unsigned int compressionLevel
```

Compression level (0-9).

0 is no compression, 9 is best compression.

7.33.2.2 interlaced

```
BOOL interlaced
```

Whether to save the PNG as interlaced.

7.33.2.3 reserved

```
unsigned int reserved[16]
```

Reserved for future use.

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

- FlyCapture2Defs_C.h

7.34 fc2PPMOption Struct Reference

Options for saving PPM images.

Data Fields

- BOOL binaryFile

  Whether to save the PPM as a binary file.
- unsigned int reserved [16]

  Reserved for future use.

7.34.1 Detailed Description

Options for saving PPM images.

7.34.2 Field Documentation

7.34.2.1 binaryFile

```
BOOL binaryFile
```

Whether to save the PPM as a binary file.

7.34.2.2 reserved

```
unsigned int reserved[16]
```

Reserved for future use.

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

- FlyCapture2Defs_C.h

7.35 fc2StrobeControl Struct Reference

A camera strobe.

Data Fields

- unsigned int source

  Source value.
- BOOL onOff

  Flag controlling on/off.
- unsigned int polarity

  Signal polarity.
- float delay

  Signal delay (in ms).
- float duration

  Signal duration (in ms).
- unsigned int reserved [8]

  Reserved for future use.

7.35.1 Detailed Description

A camera strobe.

7.35.2 Field Documentation

7.35.2.1 delay

```
float delay
```

Signal delay (in ms).

7.35.2.2 duration

```
float duration
```

Signal duration (in ms).

7.35.2.3 onOff

```
BOOL onOff
```

Flag controlling on/off.

7.35.2.4 polarity

```
unsigned int polarity
```

Signal polarity.

7.35.2.5 reserved

```
unsigned int reserved[8]
```

Reserved for future use.

7.35.2.6 source

```
unsigned int source
```

Source value.

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

- FlyCapture2Defs_C.h

7.36 fc2StrobeInfo Struct Reference

A camera strobe property.

Data Fields

- unsigned int source
    Source value.
- BOOL present
    Presence of strobe.
- BOOL readOutSupported
    Flag indicating if strobe value can be read out.
- BOOL onOffSupported
    Flag indicating if on/off is supported.
- BOOL polaritySupported
    Flag indicating if polarity is supported.
- float minValue
    Minimum value.
- float maxValue
    Maximum value.
- unsigned int reserved [8]
    Reserved for future use.

7.36.1 Detailed Description

A camera strobe property.

7.36.2 Field Documentation

7.36.2.1 maxValue

```
float maxValue
```

Maximum value.

7.36.2.2 minValue

```
float minValue
```

Minimum value.

7.36.2.3 onOffSupported

```
BOOL onOffSupported
```

Flag indicating if on/off is supported.

7.36.2.4 polaritySupported

```
BOOL polaritySupported
```

Flag indicating if polarity is supported.

7.36.2.5 present

```
BOOL present
```

Presence of strobe.

7.36.2.6 readOutSupported

```
BOOL readOutSupported
```

Flag indicating if strobe value can be read out.

7.36.2.7 reserved

```
unsigned int reserved[8]
```

Reserved for future use.

7.36.2.8 source

```
unsigned int source
```

Source value.

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

- FlyCapture2Defs_C.h

7.37 fc2SystemInfo Struct Reference

Description of the system.

Data Fields

- fc2OSType osType

  Operating system type as described by OSType.
- char osDescription [MAX_STRING_LENGTH]

  Detailed description of the operating system.
- fc2ByteOrder byteOrder

  Byte order of the system.
- size_t sysMemSize

  Amount of memory available on the system.
- char cpuDescription [MAX_STRING_LENGTH]

  Detailed description of the CPU.
- size_t numCpuCores

  Number of cores on all CPUs on the system.
- char driverList [MAX_STRING_LENGTH]

  List of drivers used.
- char libraryList [MAX_STRING_LENGTH]

  List of libraries used.
- char gpuDescription [MAX_STRING_LENGTH]

  Detailed description of the GPU.
- size_t screenWidth

  Screen resolution width in pixels.
- size_t screenHeight

  Screen resolution height in pixels.
- unsigned int reserved [16]

  Reserved for future use.

7.37.1 Detailed Description

Description of the system.

7.37.2 Field Documentation

7.37.2.1 byteOrder

```
fc2ByteOrder byteOrder
```

Byte order of the system.

7.37.2.2 cpuDescription

```
char cpuDescription[MAX_STRING_LENGTH]
```

Detailed description of the CPU.

7.37.2.3 driverList

```
char driverList[MAX_STRING_LENGTH]
```

List of drivers used.

7.37.2.4 gpuDescription

```
char gpuDescription[MAX_STRING_LENGTH]
```

Detailed description of the GPU.

7.37.2.5 libraryList

```
char libraryList[MAX_STRING_LENGTH]
```

List of libraries used.

7.37.2.6 numCpuCores

```
size_t numCpuCores
```

Number of cores on all CPUs on the system.

7.37.2.7 osDescription

```
char osDescription[MAX_STRING_LENGTH]
```

Detailed description of the operating system.

7.37.2.8 osType

```
fc2OSType osType
```

Operating system type as described by OSType.

7.37.2.9 reserved

```
unsigned int reserved[16]
```

Reserved for future use.

7.37.2.10 screenHeight

```
size_t screenHeight
```

Screen resolution height in pixels.

7.37.2.11 screenWidth

```
size_t screenWidth
```

Screen resolution width in pixels.

7.37.2.12 sysMemSize

```
size_t sysMemSize
```

Amount of memory available on the system.

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

- FlyCapture2Defs_C.h

7.38 fc2TIFFOption Struct Reference

Options for saving TIFF images.

Data Fields

- fc2TIFFCompressionMethod compression

  Compression method to use for encoding TIFF images.
- unsigned int reserved [16]

  Reserved for future use.

7.38.1 Detailed Description

Options for saving TIFF images.

7.38.2 Field Documentation

7.38.2.1 compression

```
fc2TIFFCompressionMethod compression
```

Compression method to use for encoding TIFF images.

7.38.2.2 reserved

```
unsigned int reserved[16]
```

Reserved for future use.

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

- FlyCapture2Defs_C.h

7.39 fc2TimeStamp Struct Reference

Timestamp information.

Data Fields

- long long seconds

    Seconds.
- unsigned int microSeconds

    Microseconds.
- unsigned int cycleSeconds

    1394 cycle time seconds.
- unsigned int cycleCount

    1394 cycle time count.
- unsigned int cycleOffset

    1394 cycle time offset.
- unsigned int reserved [8]

    Reserved for future use.

7.39.1 Detailed Description

Timestamp information.

7.39.2 Field Documentation

7.39.2.1 cycleCount

```
unsigned int cycleCount
```

1394 cycle time count.

7.39.2.2 cycleOffset

```
unsigned int cycleOffset
```

1394 cycle time offset.

7.39.2.3 cycleSeconds

```
unsigned int cycleSeconds
```

1394 cycle time seconds.

7.39.2.4 microSeconds

```
unsigned int microSeconds
```

Microseconds.

7.39.2.5 reserved

```
unsigned int reserved[8]
```

Reserved for future use.

7.39.2.6 seconds

```
long long seconds
```

Seconds.

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

- FlyCapture2Defs_C.h

7.40 fc2TriggerDelay Struct Reference

A specific camera property.

Data Fields

- fc2PropertyType type

  Property info type.
- BOOL present

  Flag indicating if the property is present.
- BOOL absControl

  Flag controlling absolute mode (real world units) or non-absolute mode (camera internal units).
- BOOL onePush

  Flag controlling one push.
- BOOL onOff

  Flag controlling on/off.
- BOOL autoManualMode

  Flag controlling auto.
- unsigned int valueA

  Value A (integer).
- unsigned int valueB

  Value B (integer).
- float absValue

  Floating point value.
- unsigned int reserved [8]

  Reserved for future use.

7.40.1 Detailed Description

A specific camera property.

For example, to set the gain to 12dB, set the following values:

- *type* - `GAIN`
- *absControl* - `true`
- *onePush* - `false`
- *onOff* - `true`
- *autoManualMode* - `false`
- *absValue* - `12.0`

7.40.2 Field Documentation

7.40.2.1 absControl

```
BOOL absControl
```

Flag controlling absolute mode (real world units) or non-absolute mode (camera internal units).

7.40.2.2 absValue

```
float absValue
```

Floating point value.

Used to configure properties in absolute mode.

7.40.2.3 autoManualMode

```
BOOL autoManualMode
```

Flag controlling auto.

7.40.2.4 onePush

```
BOOL onePush
```

Flag controlling one push.

7.40.2.5 onOff

```
BOOL onOff
```

Flag controlling on/off.

7.40.2.6 present

```
BOOL present
```

Flag indicating if the property is present.

7.40.2.7 reserved

```
unsigned int reserved[8]
```

Reserved for future use.

7.40.2.8 type

```
fc2PropertyType type
```

Property info type.

7.40.2.9 valueA

```
unsigned int valueA
```

Value A (integer).

Used to configure properties in non-absolute mode.

7.40.2.10 valueB

```
unsigned int valueB
```

Value B (integer).

For white balance, value B applies to the blue value and value A applies to the red value.

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

- FlyCapture2Defs_C.h

7.41 fc2TriggerDelayInfo Struct Reference

Information about a specific camera property.

Data Fields

- fc2PropertyType type
    Property info type.
- BOOL present
    Flag indicating if the property is present.
- BOOL autoSupported
    Flag indicating if auto is supported.
- BOOL manualSupported
    Flag indicating if manual is supported.
- BOOL onOffSupported
    Flag indicating if on/off is supported.
- BOOL onePushSupported
    Flag indicating if one push is supported.
- BOOL absValSupported
    Flag indicating if absolute mode is supported.

- BOOL readOutSupported

    Flag indicating if property value can be read out.
- unsigned int min

    Minimum value (as an integer).
- unsigned int max

    Maximum value (as an integer).
- float absMin

    Minimum value (as a floating point value).
- float absMax

    Maximum value (as a floating point value).
- char pUnits [MAX_STRING_LENGTH]

    Textual description of units.
- char pUnitAbbr [MAX_STRING_LENGTH]

    Abbreviated textual description of units.
- unsigned int reserved [8]

    Reserved for future use.

7.41.1 Detailed Description

Information about a specific camera property.

This structure is also also used as the TriggerDelayInfo structure.

7.41.2 Field Documentation

7.41.2.1 absMax

```
float absMax
```

Maximum value (as a floating point value).

7.41.2.2 absMin

```
float absMin
```

Minimum value (as a floating point value).

7.41.2.3 absValSupported

```
BOOL absValSupported
```

Flag indicating if absolute mode is supported.

7.41.2.4 autoSupported

```
BOOL autoSupported
```

Flag indicating if auto is supported.

7.41.2.5 manualSupported

```
BOOL manualSupported
```

Flag indicating if manual is supported.

7.41.2.6 max

```
unsigned int max
```

Maximum value (as an integer).

7.41.2.7 min

```
unsigned int min
```

Minimum value (as an integer).

7.41.2.8 onePushSupported

```
BOOL onePushSupported
```

Flag indicating if one push is supported.

7.41.2.9 onOffSupported

```
BOOL onOffSupported
```

Flag indicating if on/off is supported.

7.41.2.10 present

```
BOOL present
```

Flag indicating if the property is present.

7.41.2.11 pUnitAbbr

```
char pUnitAbbr[MAX_STRING_LENGTH]
```

Abbreviated textual description of units.

7.41.2.12 pUnits

```
char pUnits[MAX_STRING_LENGTH]
```

Textual description of units.

7.41.2.13 readOutSupported

```
BOOL readOutSupported
```

Flag indicating if property value can be read out.

7.41.2.14 reserved

```
unsigned int reserved[8]
```

Reserved for future use.

7.41.2.15 type

```
fc2PropertyType type
```

Property info type.

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

- FlyCapture2Defs_C.h

7.42 fc2TriggerMode Struct Reference

A camera trigger.

Data Fields

- BOOL onOff

  Flag controlling on/off.
- unsigned int polarity

  Polarity value.
- unsigned int source

  Source value.
- unsigned int mode

  Mode value.
- unsigned int parameter

  Parameter value.
- unsigned int reserved [8]

  Reserved for future use.

7.42.1 Detailed Description

A camera trigger.

7.42.2 Field Documentation

7.42.2.1 mode

```
unsigned int mode
```

Mode value.

7.42.2.2 onOff

```
BOOL onOff
```

Flag controlling on/off.

7.42.2.3 parameter

```
unsigned int parameter
```

Parameter value.

7.42.2.4 polarity

```
unsigned int polarity
```

Polarity value.

7.42.2.5 reserved

```
unsigned int reserved[8]
```

Reserved for future use.

7.42.2.6 source

```
unsigned int source
```

Source value.

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

- FlyCapture2Defs_C.h

7.43 fc2TriggerModeInfo Struct Reference

Information about a camera trigger property.

Data Fields

- BOOL present

  Presence of trigger mode.
- BOOL readOutSupported

  Flag indicating if trigger value can be read out.
- BOOL onOffSupported

  Flag indicating if on/off is supported.
- BOOL polaritySupported

  Flag indicating if polarity is supported.
- BOOL valueReadable

  Flag indicating if the value is readable.
- unsigned int sourceMask

  Source mask.
- BOOL softwareTriggerSupported

  Flag indicating if software trigger is supported.
- unsigned int modeMask

  Mode mask.
- unsigned int reserved [8]

  Reserved for future use.

7.43.1 Detailed Description

Information about a camera trigger property.

7.43.2 Field Documentation

7.43.2.1 modeMask

```
unsigned int modeMask
```

Mode mask.

7.43.2.2 onOffSupported

```
BOOL onOffSupported
```

Flag indicating if on/off is supported.

7.43.2.3 polaritySupported

```
BOOL polaritySupported
```

Flag indicating if polarity is supported.

7.43.2.4 present

```
BOOL present
```

Presence of trigger mode.

7.43.2.5 readOutSupported

```
BOOL readOutSupported
```

Flag indicating if trigger value can be read out.

7.43.2.6 reserved

```
unsigned int reserved[8]
```

Reserved for future use.

7.43.2.7 softwareTriggerSupported

```
BOOL softwareTriggerSupported
```

Flag indicating if software trigger is supported.

7.43.2.8 sourceMask

```
unsigned int sourceMask
```

Source mask.

7.43.2.9 valueReadable

```
BOOL valueReadable
```

Flag indicating if the value is readable.

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

- FlyCapture2Defs_C.h

7.44 fc2Version Struct Reference

The current version of the library.

Data Fields

- unsigned int major

  Major version number.
- unsigned int minor

  Minor version number.
- unsigned int type

  Type version number.
- unsigned int build

  Build version number.

7.44.1 Detailed Description

The current version of the library.

7.44.2 Field Documentation

7.44.2.1 build

```
unsigned int build
```

Build version number.

7.44.2.2 major

```
unsigned int major
```

Major version number.

7.44.2.3 minor

```
unsigned int minor
```

Minor version number.

7.44.2.4 type

```
unsigned int type
```

Type version number.

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

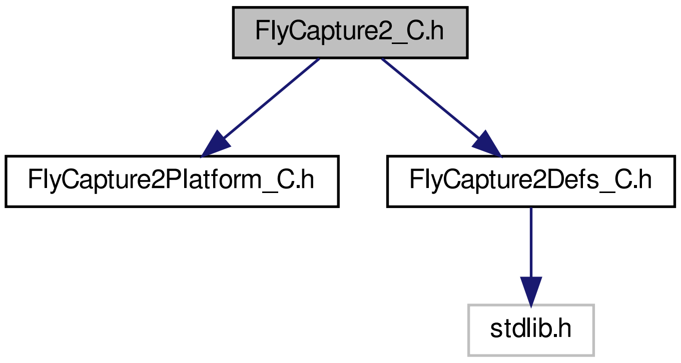
- FlyCapture2Defs_C.h

Chapter 8

File Documentation

8.1 FlyCapture2_C.h File Reference

Include dependency graph for FlyCapture2_C.h:



Functions

- FLYCAPTURE2_C_API fc2Error fc2CreateContext (fc2Context ∗pContext)

    Create a FC2 context for IIDC camera.
- FLYCAPTURE2_C_API fc2Error fc2CreateGigEContext (fc2Context ∗pContext)

    Create a FC2 context for a GigE Vision camera.
- FLYCAPTURE2_C_API fc2Error fc2DestroyContext (fc2Context context)

    Destroy the FC2 context.
- FLYCAPTURE2_C_API fc2Error fc2FireBusReset (fc2Context context, fc2PGRGuid ∗pGuid)

    Fire a bus reset.
- FLYCAPTURE2_C_API fc2Error fc2GetNumOfCameras (fc2Context context, unsigned int ∗pNumCameras)

    Gets the number of cameras attached to the PC.

- FLYCAPTURE2_C_API fc2Error fc2GetCameraFromIPAddress (fc2Context context, fc2IPAddress ip←Address, fc2PGRGuid ∗pGuid)

  Gets the PGRGuid for a camera with the specified IPv4 address.
- FLYCAPTURE2_C_API fc2Error fc2GetCameraFromIndex (fc2Context context, unsigned int index, fc2PG←RGuid ∗pGuid)

  Gets the PGRGuid for a camera on the PC.
- FLYCAPTURE2_C_API fc2Error fc2GetCameraFromSerialNumber (fc2Context context, unsigned int serial←Number, fc2PGRGuid ∗pGuid)

  Gets the PGRGuid for a camera on the PC.
- FLYCAPTURE2_C_API fc2Error fc2GetCameraSerialNumberFromIndex (fc2Context context, unsigned int index, unsigned int ∗pSerialNumber)

  Gets the serial number of the camera with the specified index.
- FLYCAPTURE2_C_API fc2Error fc2GetInterfaceTypeFromGuid (fc2Context context, fc2PGRGuid ∗pGuid, fc2InterfaceType ∗pInterfaceType)

  Gets the interface type associated with a PGRGuid.
- FLYCAPTURE2_C_API fc2Error fc2GetNumOfDevices (fc2Context context, unsigned int ∗pNumDevices)

  Gets the number of devices.
- FLYCAPTURE2_C_API fc2Error fc2GetDeviceFromIndex (fc2Context context, unsigned int index, fc2PGR←Guid ∗pGuid)

  Gets the PGRGuid for a device.
- FLYCAPTURE2_C_API fc2Error fc2ReadPhyRegister (fc2Context context, fc2PGRGuid guid, unsigned int page, unsigned int port, unsigned int address, unsigned int ∗pValue)

  Read a phy register on the specified device.
- FLYCAPTURE2_C_API fc2Error fc2WritePhyRegister (fc2Context context, fc2PGRGuid guid, unsigned int page, unsigned int port, unsigned int address, unsigned int value)

  Write a phy register on the specified device.
- FLYCAPTURE2_C_API fc2Error fc2GetUsbLinkInfo (fc2Context context, fc2PGRGuid guid, unsigned int ∗pValue)

  Read usb link info for the port that the specified device is connected to.
- FLYCAPTURE2_C_API fc2Error fc2GetUsbPortStatus (fc2Context context, fc2PGRGuid guid, unsigned int ∗pValue)

  Read usb port status for the port that the specified device is connected to.
- FLYCAPTURE2_C_API fc2Error fc2GetTopology (fc2Context context, fc2TopologyNodeContext ∗p←TopologyNodeContext)

  Gets the topology information for the PC.
- FLYCAPTURE2_C_API fc2Error fc2RegisterCallback (fc2Context context, fc2BusEventCallback enum←Callback, fc2BusCallbackType callbackType, void ∗pParameter, fc2CallbackHandle ∗pCallbackHandle)

  Register a callback function that will be called when the specified callback event occurs.
- FLYCAPTURE2_C_API fc2Error fc2UnregisterCallback (fc2Context context, fc2CallbackHandle callback←Handle)

  Unregister a callback function.
- FLYCAPTURE2_C_API fc2Error fc2RescanBus (fc2Context context)

  Force a rescan of the buses.
- FLYCAPTURE2_C_API fc2Error fc2ForceIPAddressToCamera (fc2Context context, fc2MACAddress mac←Address, fc2IPAddress ipAddress, fc2IPAddress subnetMask, fc2IPAddress defaultGateway)

  Force the camera with the specific MAC address to the specified IP address, subnet mask and default gateway.
- FLYCAPTURE2_C_API fc2Error fc2ForceAllIPAddressesAutomatically ()

  Force all cameras on the network to be assigned sequential IP addresses on the same subnet as the network adapters that they are connected to.
- FLYCAPTURE2_C_API fc2Error fc2ForceIPAddressAutomatically (unsigned int serialNumber)

  Force cameras on the network to be assigned sequential IP addresses on the same subnet as the network adapters that it is connected to.

- FLYCAPTURE2_C_API fc2Error fc2DiscoverGigECameras (fc2Context context, fc2CameraInfo ∗gigE←
Cameras, unsigned int ∗arraySize)

    Discover all cameras connected to the network even if they reside on a different subnet.
- FLYCAPTURE2_C_API fc2Error fc2IsCameraControlable (fc2Context context, fc2PGRGuid ∗pGuid, BOOL ∗pControlable)

    Query whether a GigE camera is controllable.
- FLYCAPTURE2_C_API fc2Error fc2Connect (fc2Context context, fc2PGRGuid ∗guid)

    Connects the fc2Context to the camera specified by the GUID.
- FLYCAPTURE2_C_API fc2Error fc2Disconnect (fc2Context context)

    Disconnects the fc2Context from the camera.
- FLYCAPTURE2_C_API BOOL fc2IsConnected (fc2Context context)

    Checks if the fc2Context is connected to a physical camera specified by a GUID.
- FLYCAPTURE2_C_API fc2Error fc2SetCallback (fc2Context context, fc2ImageEventCallback pCallbackFn, void ∗pCallbackData)

    Sets the callback data to be used on completion of image transfer.
- FLYCAPTURE2_C_API fc2Error fc2StartCapture (fc2Context context)

    Starts isochronous image capture.
- FLYCAPTURE2_C_API fc2Error fc2StartCaptureCallback (fc2Context context, fc2ImageEventCallback p←
CallbackFn, void ∗pCallbackData)

    Starts isochronous image capture.
- FLYCAPTURE2_C_API fc2Error fc2StartSyncCapture (unsigned int numCameras, fc2Context ∗pContexts)

    Starts synchronized isochronous image capture on multiple cameras.
- FLYCAPTURE2_C_API fc2Error fc2StartSyncCaptureCallback (unsigned int numCameras, fc2Context ∗p←
Contexts, fc2ImageEventCallback ∗pCallbackFns, void ∗∗pCallbackDataArray)

    Starts synchronized isochronous image capture on multiple cameras.
- FLYCAPTURE2_C_API fc2Error fc2RetrieveBuffer (fc2Context context, fc2Image ∗pImage)

    Retrieves the next image object containing the next image.
- FLYCAPTURE2_C_API fc2Error fc2StopCapture (fc2Context context)

    Stops isochronous image transfer and cleans up all associated resources.
- FLYCAPTURE2_C_API fc2Error fc2WaitForBufferEvent (fc2Context context, fc2Image ∗pImage, unsigned int eventNumber)

    Retrieves the next image event containing the next part of the image.
- FLYCAPTURE2_C_API fc2Error fc2SetUserBuffers (fc2Context context, unsigned char ∗const ppMem←
Buffers, int size, int nNumBuffers)

    Specify user allocated buffers to use as image data buffers.
- FLYCAPTURE2_C_API fc2Error fc2GetConfiguration (fc2Context context, fc2Config ∗config)

    Get the configuration associated with the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetConfiguration (fc2Context context, fc2Config ∗config)

    Set the configuration associated with the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetCameraInfo (fc2Context context, fc2CameraInfo ∗pCameraInfo)

    Retrieves information from the camera such as serial number, model name and other camera information.
- FLYCAPTURE2_C_API fc2Error fc2GetPropertyInfo (fc2Context context, fc2PropertyInfo ∗propInfo)

    Retrieves information about the specified camera property.
- FLYCAPTURE2_C_API fc2Error fc2GetProperty (fc2Context context, fc2Property ∗prop)

    Reads the settings for the specified property from the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetProperty (fc2Context context, fc2Property ∗prop)

    Writes the settings for the specified property to the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetPropertyBroadcast (fc2Context context, fc2Property ∗prop)

    Writes the settings for the specified property to the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetGPIOPinDirection (fc2Context context, unsigned int pin, unsigned int ∗pDirection)

    Get the GPIO pin direction for the specified pin.

- FLYCAPTURE2_C_API fc2Error fc2SetGPIOPinDirection (fc2Context context, unsigned int pin, unsigned int direction)

  Set the GPIO pin direction for the specified pin.
- FLYCAPTURE2_C_API fc2Error fc2SetGPIOPinDirectionBroadcast (fc2Context context, unsigned int pin, unsigned int direction)

  Set the GPIO pin direction for the specified pin.
- FLYCAPTURE2_C_API fc2Error fc2GetTriggerModeInfo (fc2Context context, fc2TriggerModeInfo ∗trigger←ModeInfo)

  Retrieve trigger information from the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetTriggerMode (fc2Context context, fc2TriggerMode ∗triggerMode)

  Retrieve current trigger settings from the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetTriggerMode (fc2Context context, fc2TriggerMode ∗triggerMode)

  Set the specified trigger settings to the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetTriggerModeBroadcast (fc2Context context, fc2TriggerMode ∗triggerMode)

  Set the specified trigger settings to the camera.
- FLYCAPTURE2_C_API fc2Error fc2FireSoftwareTrigger (fc2Context context)

  Fire the software trigger according to the DCAM specifications.
- FLYCAPTURE2_C_API fc2Error fc2FireSoftwareTriggerBroadcast (fc2Context context)

  Fire the software trigger according to the DCAM specifications.
- FLYCAPTURE2_C_API fc2Error fc2GetTriggerDelayInfo (fc2Context context, fc2TriggerDelayInfo ∗trigger←DelayInfo)

  Retrieve trigger delay information from the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetTriggerDelay (fc2Context context, fc2TriggerDelay ∗triggerDelay)

  Retrieve current trigger delay settings from the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetTriggerDelay (fc2Context context, fc2TriggerDelay ∗triggerDelay)

  Set the specified trigger delay settings to the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetTriggerDelayBroadcast (fc2Context context, fc2TriggerDelay ∗triggerDelay)

  Set the specified trigger delay settings to the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetStrobeInfo (fc2Context context, fc2StrobeInfo ∗strobeInfo)

  Retrieve strobe information from the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetStrobe (fc2Context context, fc2StrobeControl ∗strobeControl)

  Retrieve current strobe settings from the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetStrobe (fc2Context context, fc2StrobeControl ∗strobeControl)

  Set current strobe settings to the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetStrobeBroadcast (fc2Context context, fc2StrobeControl ∗strobe←Control)

  Set current strobe settings to the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetLUTInfo (fc2Context context, fc2LUTData ∗pData)

  Query if LUT support is available on the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetLUTBankInfo (fc2Context context, unsigned int bank, BOOL ∗p←ReadSupported, BOOL ∗pWriteSupported)

  Query the read/write status of a single LUT bank.
- FLYCAPTURE2_C_API fc2Error fc2GetActiveLUTBank (fc2Context context, unsigned int ∗pActiveBank)

  Get the LUT bank that is currently being used.
- FLYCAPTURE2_C_API fc2Error fc2SetActiveLUTBank (fc2Context context, unsigned int activeBank)

  Set the LUT bank that will be used.
- FLYCAPTURE2_C_API fc2Error fc2EnableLUT (fc2Context context, BOOL on)

  Enable or disable LUT functionality on the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetLUTChannel (fc2Context context, unsigned int bank, unsigned int channel, unsigned int sizeEntries, unsigned int ∗pEntries)

Get the LUT channel settings from the camera.

- FLYCAPTURE2_C_API fc2Error fc2SetLUTChannel (fc2Context context, unsigned int bank, unsigned int channel, unsigned int sizeEntries, unsigned int ∗pEntries)

  Set the LUT channel settings to the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetMemoryChannel (fc2Context context, unsigned int ∗pCurrent←Channel)

  Retrieve the current memory channel from the camera.
- FLYCAPTURE2_C_API fc2Error fc2SaveToMemoryChannel (fc2Context context, unsigned int channel)

  Save the current settings to the specified current memory channel.
- FLYCAPTURE2_C_API fc2Error fc2RestoreFromMemoryChannel (fc2Context context, unsigned int channel)

  Restore the specified current memory channel.
- FLYCAPTURE2_C_API fc2Error fc2GetMemoryChannelInfo (fc2Context context, unsigned int ∗pNum←Channels)

  Query the camera for memory channel support.
- FLYCAPTURE2_C_API fc2Error fc2GetEmbeddedImageInfo (fc2Context context, fc2EmbeddedImageInfo ∗pInfo)

  Get the current status of the embedded image information register, as well as the availability of each embedded property.
- FLYCAPTURE2_C_API fc2Error fc2SetEmbeddedImageInfo (fc2Context context, fc2EmbeddedImageInfo ∗pInfo)

  Sets the on/off values of the embedded image information structure to the camera.
- FLYCAPTURE2_C_API fc2Error fc2WriteRegister (fc2Context context, unsigned int address, unsigned int value)

  Write to the specified register on the camera.
- FLYCAPTURE2_C_API fc2Error fc2ReadRegister (fc2Context context, unsigned int address, unsigned int ∗pValue)

  Read the specified register from the camera.
- FLYCAPTURE2_C_API fc2Error fc2WriteRegisterBroadcast (fc2Context context, unsigned int address, unsigned int value)

  Write to the specified register on the camera with broadcast.
- FLYCAPTURE2_C_API fc2Error fc2WriteRegisterBlock (fc2Context context, unsigned short addressHigh, unsigned int addressLow, const unsigned int ∗pBuffer, unsigned int length)

  Write to the specified register block on the camera.
- FLYCAPTURE2_C_API fc2Error fc2ReadRegisterBlock (fc2Context context, unsigned short addressHigh, unsigned int addressLow, unsigned int ∗pBuffer, unsigned int length)

  Write to the specified register block on the camera.
- FLYCAPTURE2_C_API const char ∗ fc2GetRegisterString (unsigned int registerVal)

  Returns a text representation of the register value.
- FLYCAPTURE2_C_API fc2Error fc2GetCycleTime (fc2Context context, fc2TimeStamp ∗pTimeStamp)

  Get cycle time from camera.
- FLYCAPTURE2_C_API fc2Error fc2GetStats (fc2Context context, fc2CameraStats ∗pCameraStats)

  Returns the camera diagnostic information.
- FLYCAPTURE2_C_API fc2Error ResetStats ()
- FLYCAPTURE2_C_API fc2Error fc2RegisterEvent (fc2Context context, fc2EventOptions ∗pOpts)

  Register the camera to issue a custom callback function call for a specific device event.
- FLYCAPTURE2_C_API fc2Error fc2DeregisterEvent (fc2Context context, fc2EventOptions ∗pOpts)

  De-register an event previously registered with the camera.
- FLYCAPTURE2_C_API fc2Error fc2RegisterAllEvents (fc2Context context, fc2EventOptions ∗pOpts)

  Register the camera to issue a custom callback function call for a specific device event.
- FLYCAPTURE2_C_API fc2Error fc2DeregisterAllEvents (fc2Context context)
- FLYCAPTURE2_C_API fc2Error fc2GetVideoModeAndFrameRateInfo (fc2Context context, fc2VideoMode videoMode, fc2FrameRate frameRate, BOOL ∗pSupported)

Query the camera to determine if the specified video mode and frame rate is supported.

- FLYCAPTURE2_C_API fc2Error fc2GetVideoModeAndFrameRate (fc2Context context, fc2VideoMode ∗videoMode, fc2FrameRate ∗frameRate)

  Get the current video mode and frame rate from the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetVideoModeAndFrameRate (fc2Context context, fc2VideoMode videoMode, fc2FrameRate frameRate)

  Set the specified video mode and frame rate to the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetFormat7Info (fc2Context context, fc2Format7Info ∗info, BOOL ∗p←Supported)

  Retrieve the availability of Format7 custom image mode and the camera capabilities for the specified Format7 mode.
- FLYCAPTURE2_C_API fc2Error fc2ValidateFormat7Settings (fc2Context context, fc2Format7ImageSettings ∗imageSettings, BOOL ∗settingsAreValid, fc2Format7PacketInfo ∗packetInfo)

  Validates Format7ImageSettings structure and returns valid packet size information if the image settings are valid.
- FLYCAPTURE2_C_API fc2Error fc2GetFormat7Configuration (fc2Context context, fc2Format7Image←Settings ∗imageSettings, unsigned int ∗packetSize, float ∗percentage)

  Get the current Format7 configuration from the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetFormat7ConfigurationPacket (fc2Context context, fc2Format7←ImageSettings ∗imageSettings, unsigned int packetSize)

  Set the current Format7 configuration to the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetFormat7Configuration (fc2Context context, fc2Format7ImageSettings ∗imageSettings, float percentSpeed)

  Set the current Format7 configuration to the camera.
- FLYCAPTURE2_C_API fc2Error fc2WriteGVCPRegister (fc2Context context, unsigned int address, unsigned int value)

  Write a GVCP register.
- FLYCAPTURE2_C_API fc2Error fc2WriteGVCPRegisterBroadcast (fc2Context context, unsigned int address, unsigned int value)

  Write a GVCP register with broadcast.
- FLYCAPTURE2_C_API fc2Error fc2ReadGVCPRegister (fc2Context context, unsigned int address, unsigned int ∗pValue)

  Read a GVCP register.
- FLYCAPTURE2_C_API fc2Error fc2WriteGVCPRegisterBlock (fc2Context context, unsigned int address, const unsigned int ∗pBuffer, unsigned int length)

  Write a GVCP register block.
- FLYCAPTURE2_C_API fc2Error fc2ReadGVCPRegisterBlock (fc2Context context, unsigned int address, unsigned int ∗pBuffer, unsigned int length)

  Read a GVCP register block.
- FLYCAPTURE2_C_API fc2Error fc2WriteGVCPMemory (fc2Context context, unsigned int address, const unsigned char ∗pBuffer, unsigned int length)

  Write a GVCP memory block.
- FLYCAPTURE2_C_API fc2Error fc2ReadGVCPMemory (fc2Context context, unsigned int address, unsigned char ∗pBuffer, unsigned int length)

  Read a GVCP memory block.
- FLYCAPTURE2_C_API fc2Error fc2GetGigEProperty (fc2Context context, fc2GigEProperty ∗pGigEProp)

  Get the specified GigEProperty.
- FLYCAPTURE2_C_API fc2Error fc2SetGigEProperty (fc2Context context, const fc2GigEProperty ∗pGigE←Prop)

  Set the specified GigEProperty.
- FLYCAPTURE2_C_API fc2Error fc2DiscoverGigEPacketSize (fc2Context context, unsigned int ∗packetSize)

  Discover the largest packet size that works for the network link between the PC and the camera.
- FLYCAPTURE2_C_API fc2Error fc2QueryGigEImagingMode (fc2Context context, fc2Mode mode, BOOL ∗isSupported)

  Check if the particular imaging mode is supported by the camera.

- FLYCAPTURE2_C_API fc2Error fc2GetGigEImagingMode (fc2Context context, fc2Mode ∗mode)

  Get the current imaging mode on the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetGigEImagingMode (fc2Context context, fc2Mode mode)

  Set the current imaging mode to the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetGigEImageSettingsInfo (fc2Context context, fc2GigEImage↩ SettingsInfo ∗pInfo)

  Get information about the image settings possible on the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetGigEImageSettings (fc2Context context, fc2GigEImageSettings ∗p↩ ImageSettings)

  Get the current image settings on the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetGigEImageSettings (fc2Context context, const fc2GigEImageSettings ∗pImageSettings)

  Set the image settings specified to the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetGigEImageBinningSettings (fc2Context context, unsigned int ∗horz↩ BinnningValue, unsigned int ∗vertBinnningValue)

  Get the current binning settings on the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetGigEImageBinningSettings (fc2Context context, unsigned int horz↩ BinnningValue, unsigned int vertBinnningValue)

  Set the specified binning values to the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetNumStreamChannels (fc2Context context, unsigned int ∗num↩ Channels)

  Get the number of stream channels present on the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetGigEStreamChannelInfo (fc2Context context, unsigned int channel, fc2GigEStreamChannel ∗pChannel)

  Get the stream channel information for the specified channel.
- FLYCAPTURE2_C_API fc2Error fc2SetGigEStreamChannelInfo (fc2Context context, unsigned int channel, fc2GigEStreamChannel ∗pChannel)

  Set the stream channel information for the specified channel.
- FLYCAPTURE2_C_API fc2Error fc2GetGigEConfig (fc2Context context, fc2GigEConfig ∗pConfig)

  Get the current gige config on the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetGigEConfig (fc2Context context, const fc2GigEConfig ∗pConfig)

  Set the gige config specified to the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetDefaultColorProcessing (fc2ColorProcessingAlgorithm default↩ Method)

  Set the default color processing algorithm.
- FLYCAPTURE2_C_API fc2Error fc2GetDefaultColorProcessing (fc2ColorProcessingAlgorithm ∗pDefault↩ Method)

  Get the default color processing algorithm.
- FLYCAPTURE2_C_API fc2Error fc2SetDefaultOutputFormat (fc2PixelFormat format)

  Set the default output pixel format.
- FLYCAPTURE2_C_API fc2Error fc2GetDefaultOutputFormat (fc2PixelFormat ∗pFormat)

  Get the default output pixel format.
- FLYCAPTURE2_C_API fc2Error fc2DetermineBitsPerPixel (fc2PixelFormat format, unsigned int ∗pBitsPer↩ Pixel)

  Calculate the bits per pixel for the specified pixel format.
- FLYCAPTURE2_C_API fc2Error fc2CreateImage (fc2Image ∗pImage)

  Create a fc2Image.
- FLYCAPTURE2_C_API fc2Error fc2DestroyImage (fc2Image ∗image)

  Destroy the fc2Image.
- FLYCAPTURE2_C_API fc2Error fc2SetImageDimensions (fc2Image ∗pImage, unsigned int rows, unsigned int cols, unsigned int stride, fc2PixelFormat pixelFormat, fc2BayerTileFormat bayerFormat)

  Sets the dimensions of the image object.

- FLYCAPTURE2_C_API fc2Error fc2GetImageDimensions (fc2Image ∗pImage, unsigned int ∗pRows, unsigned int ∗pCols, unsigned int ∗pStride, fc2PixelFormat ∗pPixelFormat, fc2BayerTileFormat ∗pBayerFormat)

  Get the image dimensions associated with the image object.
- FLYCAPTURE2_C_API fc2Error fc2SetImageColorProcessing (fc2Image ∗pImage, fc2ColorProcessing←Algorithm colorProc)

  Set the color processing algorithm.
- FLYCAPTURE2_C_API fc2Error fc2GetImageColorProcessing (fc2Image ∗pImage, fc2ColorProcessing←Algorithm ∗pColorProc)

  Get the current color processing algorithm.
- FLYCAPTURE2_C_API fc2Error fc2SetImageData (fc2Image ∗pImage, const unsigned char ∗pData, unsigned int dataSize)

  Set the data of the Image object.
- FLYCAPTURE2_C_API fc2Error fc2GetImageData (fc2Image ∗pImage, unsigned char ∗∗ppData)

  Get a pointer to the data associated with the image.
- FLYCAPTURE2_C_API fc2Error fc2GetImageMetadata (fc2Image ∗pImage, fc2ImageMetadata ∗pImage←MetaData)

  Get the metadata associated with the image.
- FLYCAPTURE2_C_API fc2TimeStamp fc2GetImageTimeStamp (fc2Image ∗pImage)

  Get the timestamp data associated with the image.
- FLYCAPTURE2_C_API fc2Error fc2SaveImage (fc2Image ∗pImage, const char ∗pFilename, fc2ImageFile←Format format)

  Save the image to the specified file name with the file format specified.
- FLYCAPTURE2_C_API fc2Error fc2SaveImageWithOption (fc2Image ∗pImage, const char ∗pFilename, fc2ImageFileFormat format, void ∗pOption)

  Save the image to the specified file name with the file format specified.
- FLYCAPTURE2_C_API fc2Error fc2ConvertImage (fc2Image ∗pImageIn, fc2Image ∗pImageOut)
- FLYCAPTURE2_C_API fc2Error fc2ConvertImageTo (fc2PixelFormat format, fc2Image ∗pImageIn, fc2Image ∗pImageOut)

  Converts the current image buffer to the specified output format and stores the result in the specified image.
- FLYCAPTURE2_C_API fc2Error fc2CalculateImageStatistics (fc2Image ∗pImage, fc2ImageStatisticsContext ∗pImageStatisticsContext)

  Calculate statistics associated with the image.
- FLYCAPTURE2_C_API fc2Error fc2CreateImageStatistics (fc2ImageStatisticsContext ∗pImageStatistics←Context)

  Create a statistics context.
- FLYCAPTURE2_C_API fc2Error fc2DestroyImageStatistics (fc2ImageStatisticsContext imageStatistics←Context)

  Destroy a statistics context.
- FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableAll (fc2ImageStatisticsContext imageStatistics←Context)

  Enable all channels.
- FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsDisableAll (fc2ImageStatisticsContext imageStatistics←Context)

  Disable all channels.
- FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableGreyOnly (fc2ImageStatisticsContext image←StatisticsContext)

  Enable only the grey channel.
- FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableRGBOnly (fc2ImageStatisticsContext image←StatisticsContext)

  Enable only the RGB channels.
- FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableHSLOnly (fc2ImageStatisticsContext image←StatisticsContext)

  Enable only the HSL channels.

- FLYCAPTURE2_C_API fc2Error fc2GetChannelStatus (fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, BOOL ∗pEnabled)

  Get the status of a statistics channel.
- FLYCAPTURE2_C_API fc2Error fc2SetChannelStatus (fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, BOOL enabled)

  Set the status of a statistics channel.
- FLYCAPTURE2_C_API fc2Error fc2GetChannelRange (fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, unsigned int ∗pMin, unsigned int ∗pMax)

  Get the range of a statistics channel.
- FLYCAPTURE2_C_API fc2Error fc2GetChannelPixelValueRange (fc2ImageStatisticsContext image←StatisticsContext, fc2StatisticsChannel channel, unsigned int ∗pPixelValueMin, unsigned int ∗pPixel←ValueMax)

  Get the range of a statistics channel.
- FLYCAPTURE2_C_API fc2Error fc2GetChannelNumPixelValues (fc2ImageStatisticsContext image←StatisticsContext, fc2StatisticsChannel channel, unsigned int ∗pNumPixelValues)

  Get the number of unique pixel values in the image.
- FLYCAPTURE2_C_API fc2Error fc2GetChannelMean (fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, float ∗pPixelValueMean)

  Get the mean of the image.
- FLYCAPTURE2_C_API fc2Error fc2GetChannelHistogram (fc2ImageStatisticsContext imageStatistics←Context, fc2StatisticsChannel channel, int ∗∗ppHistogram)

  Get the histogram for the image.
- FLYCAPTURE2_C_API fc2Error fc2GetImageStatistics (fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, unsigned int ∗pRangeMin, unsigned int ∗pRangeMax, unsigned int ∗p←PixelValueMin, unsigned int ∗pPixelValueMax, unsigned int ∗pNumPixelValues, float ∗pPixelValueMean, int ∗∗ppHistogram)

  Get all statistics for the image.
- FLYCAPTURE2_C_API fc2Error fc2CreateTopologyNode (fc2TopologyNodeContext ∗pTopologyNode←Context)

  Create a TopologyNode context.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetGuid (fc2TopologyNodeContext TopologyNode←Context, fc2PGRGuid ∗pGuid)

  Get the PGRGuid associated with the node.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetDeviceId (fc2TopologyNodeContext TopologyNode←Context, int ∗pID)

  Get the device ID associated with the node.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetNodeType (fc2TopologyNodeContext TopologyNode←Context, fc2NodeType ∗pNodeType)

  Get the node type associated with the node.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetInterfaceType (fc2TopologyNodeContext Topology←NodeContext, fc2InterfaceType ∗pInterfaceType)

  Get the interface type associated with the node.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetNumChildren (fc2TopologyNodeContext Topology←NodeContext, unsigned int ∗pNumChildNodes)

  Get the number of child nodes.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetChild (fc2TopologyNodeContext TopologyNode←Context, unsigned int position, fc2TopologyNodeContext ∗pChildTopologyNodeContext)

  Get child node located at the specified position.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeAddChild (fc2TopologyNodeContext TopologyNode←Context, fc2TopologyNodeContext TopologyNodeChildContext)

  Add the specified TopologyNode as a child of the node.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetNumPorts (fc2TopologyNodeContext TopologyNode←Context, unsigned int ∗pNumPorts)

Get the number of ports.

- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetPortType (fc2TopologyNodeContext TopologyNode←Context, unsigned int position, fc2PortType ∗pPortType)

  Get type of port located at the specified position.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeAddPortType (fc2TopologyNodeContext TopologyNode←Context, fc2PortType portType)

  Add the specified PortType as a port of the node.
- FLYCAPTURE2_C_API BOOL fc2TopologyNodeAssignGuidToNode (fc2TopologyNodeContext Topology←NodeContext, fc2PGRGuid guid, int deviceId)

  Assign a PGRGuid and device ID to the node.
- FLYCAPTURE2_C_API BOOL fc2TopologyNodeAssignGuidToNodeEx (fc2TopologyNodeContext Topology←NodeContext, fc2PGRGuid guid, int deviceId, fc2NodeType nodeType)

  Assign a PGRGuid, device ID and nodeType to the node.
- FLYCAPTURE2_C_API fc2Error fc2DestroyTopologyNode (fc2TopologyNodeContext TopologyNodeContext)

  Destroy a TopologyNode context.
- FLYCAPTURE2_C_API fc2Error fc2CheckDriver (const fc2PGRGuid ∗pGuid)

  Check for driver compatibility for the given camera guid.
- FLYCAPTURE2_C_API fc2Error fc2GetDriverDeviceName (const fc2PGRGuid ∗pGuid, char ∗pDevice←Name, size_t ∗deviceNameLength)

  Get the driver's name for a device.
- FLYCAPTURE2_C_API fc2Error fc2GetSystemInfo (fc2SystemInfo ∗pSystemInfo)

  Get system information.
- FLYCAPTURE2_C_API fc2Error fc2GetLibraryVersion (fc2Version ∗pVersion)

  Get library version.
- FLYCAPTURE2_C_API fc2Error fc2LaunchBrowser (const char ∗pAddress)

  Launch a URL in the system default browser.
- FLYCAPTURE2_C_API fc2Error fc2LaunchHelp (const char ∗pFileName)

  Open a CHM file in the system default CHM viewer.
- FLYCAPTURE2_C_API fc2Error fc2LaunchCommand (const char ∗pCommand)

  Execute a command in the terminal.
- FLYCAPTURE2_C_API fc2Error fc2LaunchCommandAsync (const char ∗pCommand, fc2AsyncCommand←Callback pCallback, void ∗pUserData)

  Execute a command in the terminal.
- FLYCAPTURE2_C_API const char ∗ fc2ErrorToDescription (fc2Error error)

  Get a string representation of an error.

8.1.1 Function Documentation

8.1.1.1 fc2CreateContext()

```
FLYCAPTURE2_C_API fc2Error fc2CreateContext (
                fc2Context * pContext )
```

Create a FC2 context for IIDC camera.

This call must be made before any other calls that use a context will succeed.

See also

fc2DestroyContext()

Parameters

pContext	A pointer to the fc2Context to be created.

Returns

A fc2Error indicating the success or failure of the function.

8.1.1.2 fc2CreateGigEContext()

```
FLYCAPTURE2_C_API fc2Error fc2CreateGigEContext (
             fc2Context * pContext )
```

Create a FC2 context for a GigE Vision camera.

This call must be made before any other calls that use a context will succeed.

See also

fc2DestroyContext()

Parameters

pContext	A pointer to the fc2Context to be created.

Returns

A fc2Error indicating the success or failure of the function.

8.1.1.3 fc2DeregisterAllEvents()

```
FLYCAPTURE2_C_API fc2Error fc2DeregisterAllEvents (
             fc2Context context )
```

8.1.1.4 fc2DeregisterEvent()

```
FLYCAPTURE2_C_API fc2Error fc2DeregisterEvent (
             fc2Context context,
             fc2EventOptions * pOpts )
```

De-register an event previously registered with the camera.

Parameters

context	The fc2Context to be used.
pOpts	Pointer to the EventOptions structure which defines the callback function to use, the event for which to register the device, and a pointer to user data (optional) to be passed to the callback function. The callback function and user data elements of the EventOptions structure are ignored in this call, and just the event name within the structure is used with this function call.

Returns

An Error indicating the success or failure of the function.

8.1.1.5 fc2DestroyContext()

```
FLYCAPTURE2_C_API fc2Error fc2DestroyContext (
            fc2Context context )
```

Destroy the FC2 context.

This must be called when the user is finished with the context in order to prevent memory leaks.

See also

fc2CreateContext()

Parameters

context	The context to be destroyed.

Returns

A fc2Error indicating the success or failure of the function.

8.1.1.6 fc2GetCycleTime()

```
FLYCAPTURE2_C_API fc2Error fc2GetCycleTime (
            fc2Context context,
            fc2TimeStamp * pTimeStamp )
```

Get cycle time from camera.

Parameters

context	The fc2Context to be used.
Timestamp	struct.

Returns

A fc2Error indicating the success or failure of the function.

8.1.1.7 fc2GetStats()

```
FLYCAPTURE2_C_API fc2Error fc2GetStats (
            fc2Context context,
            fc2CameraStats * pCameraStats )
```

Returns the camera diagnostic information.

Parameters

context	The fc2Context to be used.
pCameraStats	Pointer to the fc2CameraStats structure.

Returns

A fc2Error indicating the success or failure of the function.

8.1.1.8 fc2RegisterAllEvents()

```
FLYCAPTURE2_C_API fc2Error fc2RegisterAllEvents (
            fc2Context context,
            fc2EventOptions * pOpts )
```

Register the camera to issue a custom callback function call for a specific device event.

Parameters

context	The fc2Context to be used.
pOpts	Pointer to the EventOptions structure which defines the callback function to use, the event for which to register the device, and a pointer to user data (optional) to be passed to the callback function. The event name element of the structure is ignored with this function call. If a single event has already been registered via RegisterEvent(), this call will fail, as the user could accidentally change the the internal callback function pointer for a queued event. The user will need to de-register all registered events, then call this function again.

Returns

An Error indicating the success or failure of the function.

8.1.1.9 fc2RegisterEvent()

```
FLYCAPTURE2_C_API fc2Error fc2RegisterEvent (
            fc2Context context,
            fc2EventOptions * pOpts )
```

Register the camera to issue a custom callback function call for a specific device event.

Parameters

context	The fc2Context to be used.
pOpts	Pointer to the EventOptions structure which defines the callback function to use, the event for which to register the device, and a pointer to user data (optional) to be passed to the callback function.

Returns

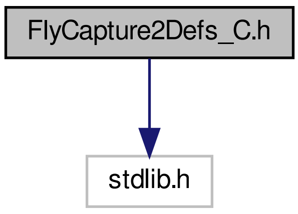
An Error indicating the success or failure of the function.

8.1.1.10 ResetStats()

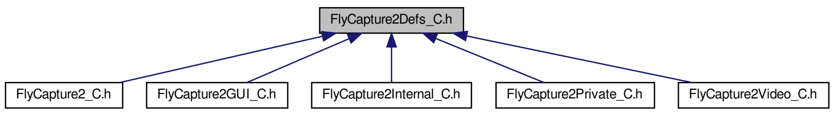
```
FLYCAPTURE2_C_API fc2Error ResetStats ( )
```

8.2 FlyCapture2Defs_C.h File Reference

Include dependency graph for FlyCapture2Defs_C.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct fc2PGRGuid

  A GUID to the camera.
- struct fc2Image
- struct fc2SystemInfo

  Description of the system.
- struct fc2Version

  The current version of the library.
- struct fc2IPAddress

  IPv4 address.
- struct fc2MACAddress

  MAC address.
- struct fc2GigEProperty

  A GigE property.
- struct fc2GigEStreamChannel

  Information about a single GigE stream channel.
- struct fc2GigEConfig

  Configuration for a GigE camera.
- struct fc2GigEImageSettingsInfo

  Format 7 information for a single mode.
- struct fc2GigEImageSettings

  Image settings for a GigE camera.
- struct fc2Format7ImageSettings

  Format 7 image settings.
- struct fc2Format7Info

  Format 7 information for a single mode.
- struct fc2Format7PacketInfo

  Format 7 packet information.
- struct fc2Config

  Configuration for a camera.
- struct fc2TriggerDelayInfo

  Information about a specific camera property.
- struct fc2TriggerDelay

  A specific camera property.
- struct fc2TriggerModeInfo

  Information about a camera trigger property.
- struct fc2TriggerMode

  A camera trigger.
- struct fc2StrobeInfo

  A camera strobe property.
- struct fc2StrobeControl

  A camera strobe.
- struct fc2TimeStamp

  Timestamp information.
- struct fc2ConfigROM

  Camera configuration ROM.
- struct fc2CameraInfo

  Camera information.
- struct fc2EmbeddedImageInfoProperty

  Properties of a single embedded image info property.

- struct fc2EmbeddedImageInfo

  Properties of the possible embedded image information.
- struct fc2ImageMetadata

  Metadata related to an image.
- struct fc2LUTData

  Information about the camera's look up table.
- struct fc2CameraStats

  Camera diagnostic information.
- struct fc2PNGOption

  Options for saving PNG images.
- struct fc2PPMOption

  Options for saving PPM images.
- struct fc2PGMOption

  Options for saving PGM images.
- struct fc2TIFFOption

  Options for saving TIFF images.
- struct fc2JPEGOption

  Options for saving JPEG image.
- struct fc2JPG2Option

  Options for saving JPEG2000 image.
- struct fc2BMPOption

  Options for saving Bitmap image.
- struct fc2EventOptions

  Options for enabling device event registration.
- struct fc2EventCallbackData

Macros

- #define FALSE 0
- #define TRUE 1
- #define FULL_32BIT_VALUE 0x7FFFFFFF
- #define MAX_STRING_LENGTH 512

Typedefs

- typedef int BOOL
- typedef void ∗ fc2Context

  A context to the FlyCapture2 C library.
- typedef void ∗ fc2GuiContext

  A context to the FlyCapture2 C GUI library.
- typedef void ∗ fc2ImageImpl

  An internal pointer used in the fc2Image structure.
- typedef void ∗ fc2ImageStatisticsContext

  A context referring to the ImageStatistics object.
- typedef void ∗ fc2TopologyNodeContext

  A context referring to the TopologyNode object.
- typedef void ∗ fc2CallbackHandle
- typedef void(∗ fc2BusEventCallback) (void ∗pParameter, unsigned int serialNumber)
- typedef void(∗ fc2ImageEventCallback) (fc2Image ∗image, void ∗pCallbackData)
- typedef void(∗ fc2AsyncCommandCallback) (fc2Error retError, void ∗pUserData)
- typedef void(∗ fc2CameraEventCallback) (void ∗pCallbackData)

Enumerations

- enum fc2Error {
  FC2_ERROR_UNDEFINED = -1,
  FC2_ERROR_OK,
  FC2_ERROR_FAILED,
  FC2_ERROR_NOT_IMPLEMENTED,
  FC2_ERROR_FAILED_BUS_MASTER_CONNECTION,
  FC2_ERROR_NOT_CONNECTED,
  FC2_ERROR_INIT_FAILED,
  FC2_ERROR_NOT_INTITIALIZED,
  FC2_ERROR_INVALID_PARAMETER,
  FC2_ERROR_INVALID_SETTINGS,
  FC2_ERROR_INVALID_BUS_MANAGER,
  FC2_ERROR_MEMORY_ALLOCATION_FAILED,
  FC2_ERROR_LOW_LEVEL_FAILURE,
  FC2_ERROR_NOT_FOUND,
  FC2_ERROR_FAILED_GUID,
  FC2_ERROR_INVALID_PACKET_SIZE,
  FC2_ERROR_INVALID_MODE,
  FC2_ERROR_NOT_IN_FORMAT7,
  FC2_ERROR_NOT_SUPPORTED,
  FC2_ERROR_TIMEOUT,
  FC2_ERROR_BUS_MASTER_FAILED,
  FC2_ERROR_INVALID_GENERATION,
  FC2_ERROR_LUT_FAILED,
  FC2_ERROR_IIDC_FAILED,
  FC2_ERROR_STROBE_FAILED,
  FC2_ERROR_TRIGGER_FAILED,
  FC2_ERROR_PROPERTY_FAILED,
  FC2_ERROR_PROPERTY_NOT_PRESENT,
  FC2_ERROR_REGISTER_FAILED,
  FC2_ERROR_READ_REGISTER_FAILED,
  FC2_ERROR_WRITE_REGISTER_FAILED,
  FC2_ERROR_ISOCH_FAILED,
  FC2_ERROR_ISOCH_ALREADY_STARTED,
  FC2_ERROR_ISOCH_NOT_STARTED,
  FC2_ERROR_ISOCH_START_FAILED,
  FC2_ERROR_ISOCH_RETRIEVE_BUFFER_FAILED,
  FC2_ERROR_ISOCH_STOP_FAILED,
  FC2_ERROR_ISOCH_SYNC_FAILED,
  FC2_ERROR_ISOCH_BANDWIDTH_EXCEEDED,
  FC2_ERROR_IMAGE_CONVERSION_FAILED,
  FC2_ERROR_IMAGE_LIBRARY_FAILURE,
  FC2_ERROR_BUFFER_TOO_SMALL,
  FC2_ERROR_IMAGE_CONSISTENCY_ERROR,
  FC2_ERROR_INCOMPATIBLE_DRIVER,
  FC2_ERROR_FORCE_32BITS = FULL_32BIT_VALUE }

  The error types returned by functions.
- enum fc2BusCallbackType {
  FC2_BUS_RESET,
  FC2_ARRIVAL,
  FC2_REMOVAL,
  FC2_CALLBACK_TYPE_FORCE_32BITS = FULL_32BIT_VALUE }

  The type of bus callback to register a callback function for.
- enum fc2GrabMode {
  FC2_DROP_FRAMES,
  FC2_BUFFER_FRAMES,

FC2_UNSPECIFIED_GRAB_MODE,
FC2_GRAB_MODE_FORCE_32BITS = FULL_32BIT_VALUE }

The grab strategy employed during image transfer.

- enum fc2GrabTimeout {
FC2_TIMEOUT_NONE = 0,
FC2_TIMEOUT_INFINITE = -1,
FC2_TIMEOUT_UNSPECIFIED = -2,
FC2_GRAB_TIMEOUT_FORCE_32BITS = FULL_32BIT_VALUE }

  Timeout options for grabbing images.
- enum fc2BandwidthAllocation {
FC2_BANDWIDTH_ALLOCATION_OFF = 0,
FC2_BANDWIDTH_ALLOCATION_ON = 1,
FC2_BANDWIDTH_ALLOCATION_UNSUPPORTED = 2,
FC2_BANDWIDTH_ALLOCATION_UNSPECIFIED = 3,
FC2_BANDWIDTH_ALLOCATION_FORCE_32BITS = FULL_32BIT_VALUE }

  Bandwidth allocation options for 1394 devices.
- enum fc2InterfaceType {
FC2_INTERFACE_IEEE1394,
FC2_INTERFACE_USB_2,
FC2_INTERFACE_USB_3,
FC2_INTERFACE_GIGE,
FC2_INTERFACE_UNKNOWN,
FC2_INTERFACE_TYPE_FORCE_32BITS = FULL_32BIT_VALUE }

  Interfaces that a camera may use to communicate with a host.
- enum fc2PropertyType {
FC2_BRIGHTNESS,
FC2_AUTO_EXPOSURE,
FC2_SHARPNESS,
FC2_WHITE_BALANCE,
FC2_HUE,
FC2_SATURATION,
FC2_GAMMA,
FC2_IRIS,
FC2_FOCUS,
FC2_ZOOM,
FC2_PAN,
FC2_TILT,
FC2_SHUTTER,
FC2_GAIN,
FC2_TRIGGER_MODE,
FC2_TRIGGER_DELAY,
FC2_FRAME_RATE,
FC2_TEMPERATURE,
FC2_UNSPECIFIED_PROPERTY_TYPE,
FC2_PROPERTY_TYPE_FORCE_32BITS = FULL_32BIT_VALUE }

  Camera properties.
- enum fc2FrameRate {
FC2_FRAMERATE_1_875,
FC2_FRAMERATE_3_75,
FC2_FRAMERATE_7_5,
FC2_FRAMERATE_15,
FC2_FRAMERATE_30,
FC2_FRAMERATE_60,
FC2_FRAMERATE_120,
FC2_FRAMERATE_240,
FC2_FRAMERATE_FORMAT7,

FC2_NUM_FRAMERATES,
FC2_FRAMERATE_FORCE_32BITS = FULL_32BIT_VALUE }

Frame rates in frames per second.

- enum fc2VideoMode {
FC2_VIDEOMODE_160x120YUV444,
FC2_VIDEOMODE_320x240YUV422,
FC2_VIDEOMODE_640x480YUV411,
FC2_VIDEOMODE_640x480YUV422,
FC2_VIDEOMODE_640x480RGB,
FC2_VIDEOMODE_640x480Y8,
FC2_VIDEOMODE_640x480Y16,
FC2_VIDEOMODE_800x600YUV422,
FC2_VIDEOMODE_800x600RGB,
FC2_VIDEOMODE_800x600Y8,
FC2_VIDEOMODE_800x600Y16,
FC2_VIDEOMODE_1024x768YUV422,
FC2_VIDEOMODE_1024x768RGB,
FC2_VIDEOMODE_1024x768Y8,
FC2_VIDEOMODE_1024x768Y16,
FC2_VIDEOMODE_1280x960YUV422,
FC2_VIDEOMODE_1280x960RGB,
FC2_VIDEOMODE_1280x960Y8,
FC2_VIDEOMODE_1280x960Y16,
FC2_VIDEOMODE_1600x1200YUV422,
FC2_VIDEOMODE_1600x1200RGB,
FC2_VIDEOMODE_1600x1200Y8,
FC2_VIDEOMODE_1600x1200Y16,
FC2_VIDEOMODE_FORMAT7,
FC2_NUM_VIDEOMODES,
FC2_VIDEOMODE_FORCE_32BITS = FULL_32BIT_VALUE }

DCAM video modes.

- enum fc2Mode {
FC2_MODE_0 = 0,
FC2_MODE_1,
FC2_MODE_2,
FC2_MODE_3,
FC2_MODE_4,
FC2_MODE_5,
FC2_MODE_6,
FC2_MODE_7,
FC2_MODE_8,
FC2_MODE_9,
FC2_MODE_10,
FC2_MODE_11,
FC2_MODE_12,
FC2_MODE_13,
FC2_MODE_14,
FC2_MODE_15,
FC2_MODE_16,
FC2_MODE_17,
FC2_MODE_18,
FC2_MODE_19,
FC2_MODE_20,
FC2_MODE_21,
FC2_MODE_22,
FC2_MODE_23,
FC2_MODE_24,
FC2_MODE_25,

FC2_MODE_26,
FC2_MODE_27,
FC2_MODE_28,
FC2_MODE_29,
FC2_MODE_30,
FC2_MODE_31,
FC2_NUM_MODES,
FC2_MODE_FORCE_32BITS = FULL_32BIT_VALUE }

Camera modes for DCAM formats as well as Format7.

- enum fc2PixelFormat {
FC2_PIXEL_FORMAT_MONO8 = 0x80000000,
FC2_PIXEL_FORMAT_411YUV8 = 0x40000000,
FC2_PIXEL_FORMAT_422YUV8 = 0x20000000,
FC2_PIXEL_FORMAT_444YUV8 = 0x10000000,
FC2_PIXEL_FORMAT_RGB8 = 0x08000000,
FC2_PIXEL_FORMAT_MONO16 = 0x04000000,
FC2_PIXEL_FORMAT_RGB16 = 0x02000000,
FC2_PIXEL_FORMAT_S_MONO16 = 0x01000000,
FC2_PIXEL_FORMAT_S_RGB16 = 0x00800000,
FC2_PIXEL_FORMAT_RAW8 = 0x00400000,
FC2_PIXEL_FORMAT_RAW16 = 0x00200000,
FC2_PIXEL_FORMAT_MONO12 = 0x00100000,
FC2_PIXEL_FORMAT_RAW12 = 0x00080000,
FC2_PIXEL_FORMAT_BGR = 0x80000008,
FC2_PIXEL_FORMAT_BGRU = 0x40000008,
FC2_PIXEL_FORMAT_RGB = FC2_PIXEL_FORMAT_RGB8,
FC2_PIXEL_FORMAT_RGBU = 0x40000002,
FC2_PIXEL_FORMAT_BGR16 = 0x02000001,
FC2_PIXEL_FORMAT_BGRU16 = 0x02000002,
FC2_PIXEL_FORMAT_422YUV8_JPEG = 0x40000001,
FC2_NUM_PIXEL_FORMATS = 20,
FC2_UNSPECIFIED_PIXEL_FORMAT = 0 }

Pixel formats available for Format7 modes.

- enum fc2BusSpeed {
FC2_BUSSPEED_S100,
FC2_BUSSPEED_S200,
FC2_BUSSPEED_S400,
FC2_BUSSPEED_S480,
FC2_BUSSPEED_S800,
FC2_BUSSPEED_S1600,
FC2_BUSSPEED_S3200,
FC2_BUSSPEED_S5000,
FC2_BUSSPEED_10BASE_T,
FC2_BUSSPEED_100BASE_T,
FC2_BUSSPEED_1000BASE_T,
FC2_BUSSPEED_10000BASE_T,
FC2_BUSSPEED_S_FASTEST,
FC2_BUSSPEED_ANY,
FC2_BUSSPEED_SPEED_UNKNOWN = -1,
FC2_BUSSPEED_FORCE_32BITS = FULL_32BIT_VALUE }

Bus speeds.

- enum fc2PCIeBusSpeed {
FC2_PCIE_BUSSPEED_2_5,
FC2_PCIE_BUSSPEED_5_0,
FC2_PCIE_BUSSPEED_UNKNOWN = -1,
FC2_PCIE_BUSSPEED_FORCE_32BITS = FULL_32BIT_VALUE }
- enum fc2DriverType {
FC2_DRIVER_1394_CAM,

FC2_DRIVER_1394_PRO,
FC2_DRIVER_1394_JUJU,
FC2_DRIVER_1394_VIDEO1394,
FC2_DRIVER_1394_RAW1394,
FC2_DRIVER_USB_NONE,
FC2_DRIVER_USB_CAM,
FC2_DRIVER_USB3_PRO,
FC2_DRIVER_GIGE_NONE,
FC2_DRIVER_GIGE_FILTER,
FC2_DRIVER_GIGE_PRO,
FC2_DRIVER_GIGE_LWF,
FC2_DRIVER_UNKNOWN = -1,
FC2_DRIVER_FORCE_32BITS = FULL_32BIT_VALUE }

Types of low level drivers that FlyCapture uses.

- enum fc2ColorProcessingAlgorithm {
FC2_DEFAULT,
FC2_NO_COLOR_PROCESSING,
FC2_NEAREST_NEIGHBOR_FAST,
FC2_EDGE_SENSING,
FC2_HQ_LINEAR,
FC2_RIGOROUS,
FC2_IPP,
FC2_DIRECTIONAL,
FC2_WEIGHTED_DIRECTIONAL,
FC2_COLOR_PROCESSING_ALGORITHM_FORCE_32BITS = FULL_32BIT_VALUE }

Color processing algorithms.

- enum fc2BayerTileFormat {
FC2_BT_NONE,
FC2_BT_RGGB,
FC2_BT_GRBG,
FC2_BT_GBRG,
FC2_BT_BGGR,
FC2_BT_FORCE_32BITS = FULL_32BIT_VALUE }

Bayer tile formats.

- enum fc2ImageFileFormat {
FC2_FROM_FILE_EXT = -1,
FC2_PGM,
FC2_PPM,
FC2_BMP,
FC2_JPEG,
FC2_JPEG2000,
FC2_TIFF,
FC2_PNG,
FC2_RAW,
FC2_IMAGE_FILE_FORMAT_FORCE_32BITS = FULL_32BIT_VALUE }

File formats to be used for saving images to disk.

- enum fc2GigEPropertyType {
FC2_HEARTBEAT,
FC2_HEARTBEAT_TIMEOUT,
PACKET_SIZE,
PACKET_DELAY }

Possible properties that can be queried from the camera.

- enum fc2StatisticsChannel {
FC2_STATISTICS_GREY,
FC2_STATISTICS_RED,
FC2_STATISTICS_GREEN,
FC2_STATISTICS_BLUE,

FC2_STATISTICS_HUE,
FC2_STATISTICS_SATURATION,
FC2_STATISTICS_LIGHTNESS,
FC2_STATISTICS_FORCE_32BITS = FULL_32BIT_VALUE }

Channels that allow statistics to be calculated.

- enum fc2OSType {
FC2_WINDOWS_X86,
FC2_WINDOWS_X64,
FC2_LINUX_X86,
FC2_LINUX_X64,
FC2_MAC,
FC2_UNKNOWN_OS,
FC2_OSTYPE_FORCE_32BITS = FULL_32BIT_VALUE }

  Possible operating systems.
- enum fc2ByteOrder {
FC2_BYTE_ORDER_LITTLE_ENDIAN,
FC2_BYTE_ORDER_BIG_ENDIAN,
FC2_BYTE_ORDER_FORCE_32BITS = FULL_32BIT_VALUE }

  Possible byte orders.
- enum fc2PortType {
NOT_CONNECTED = 1,
CONNECTED_TO_PARENT,
CONNECTED_TO_CHILD }

  Possible states of a port on a node.
- enum fc2NodeType {
COMPUTER,
BUS,
CAMERA,
NODE }

  Type of node.
- enum fc2TIFFCompressionMethod {
FC2_TIFF_NONE = 1,
FC2_TIFF_PACKBITS,
FC2_TIFF_DEFLATE,
FC2_TIFF_ADOBE_DEFLATE,
FC2_TIFF_CCITTFAX3,
FC2_TIFF_CCITTFAX4,
FC2_TIFF_LZW,
FC2_TIFF_JPEG }

8.2.1 Enumeration Type Documentation

8.2.1.1 fc2ByteOrder

```
enum fc2ByteOrder
```

Possible byte orders.

Enumerator

FC2_BYTE_ORDER_LITTLE_ENDIAN	
FC2_BYTE_ORDER_BIG_ENDIAN	
FC2_BYTE_ORDER_FORCE_32BITS	

8.2.1.2 fc2NodeType

```
enum fc2NodeType
```

Type of node.

Enumerator

COMPUTER	
BUS	
CAMERA	
NODE	

8.2.1.3 fc2OSType

```
enum fc2OSType
```

Possible operating systems.

Enumerator

FC2_WINDOWS_X86	All Windows 32-bit variants.
FC2_WINDOWS_X64	All Windows 64-bit variants.
FC2_LINUX_X86	All Linux 32-bit variants.
FC2_LINUX_X64	All Linux 32-bit variants.
FC2_MAC	Mac OSX.
FC2_UNKNOWN_OS	Unknown operating system.
FC2_OSTYPE_FORCE_32BITS	

8.2.1.4 fc2PortType

```
enum fc2PortType
```

Possible states of a port on a node.

Enumerator

NOT_CONNECTED	
CONNECTED_TO_PARENT	
CONNECTED_TO_CHILD	

8.2.1.5 fc2StatisticsChannel

```
enum fc2StatisticsChannel
```

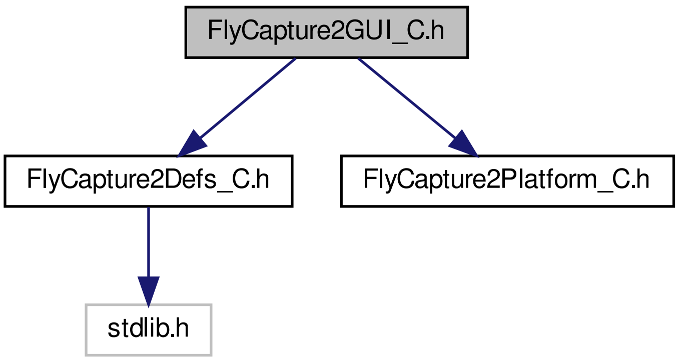
Channels that allow statistics to be calculated.

Enumerator

FC2_STATISTICS_GREY	
FC2_STATISTICS_RED	
FC2_STATISTICS_GREEN	
FC2_STATISTICS_BLUE	
FC2_STATISTICS_HUE	
FC2_STATISTICS_SATURATION	
FC2_STATISTICS_LIGHTNESS	
FC2_STATISTICS_FORCE_32BITS	

8.3 FlyCapture2GUI_C.h File Reference

Include dependency graph for FlyCapture2GUI_C.h:



Functions

- FLYCAPTURE2_C_API fc2Error fc2CreateGUIContext (fc2GuiContext ∗pContext)

  Create a GUI context.
- FLYCAPTURE2_C_API fc2Error fc2DestroyGUIContext (fc2GuiContext context)

  Destroy a GUI context.
- FLYCAPTURE2_C_API void fc2GUIConnect (fc2GuiContext context, fc2Context cameraContext)

Connect GUI context to a camera context.

- FLYCAPTURE2_C_API void fc2GUIDisconnect (fc2GuiContext context)

  Disconnect GUI context from camera.
- FLYCAPTURE2_C_API void fc2Disonnect (fc2GuiContext context) __attribute__((deprecated))

  Disconnect GUI context from camera.
- FLYCAPTURE2_C_API void fc2Show (fc2GuiContext context)

  Show the GUI.
- FLYCAPTURE2_C_API void fc2Hide (fc2GuiContext context)

  Hide the GUI.
- FLYCAPTURE2_C_API BOOL fc2IsVisible (fc2GuiContext context)

  Check if the GUI is visible.
- FLYCAPTURE2_C_API void fc2ShowModal (fc2GuiContext context, BOOL ∗pOkSelected, fc2PGRGuid ∗guidArray, unsigned int ∗size)

  Show the camera selection dialog.

8.3.1 Function Documentation

8.3.1.1 fc2CreateGUIContext()

```
FLYCAPTURE2_C_API fc2Error fc2CreateGUIContext (
            fc2GuiContext ∗ pContext )
```

Create a GUI context.

Any GigE cameras that were connected prior to this call will lose CCP after the call. Consider creating a GUI context prior to connecting any GigE cameras or calling connect on any outstanding GigE camera context.

Parameters

pContext	Pointer to context to be created.

Returns

An Error indicating the success or failure of the function.

8.3.1.2 fc2DestroyGUIContext()

```
FLYCAPTURE2_C_API fc2Error fc2DestroyGUIContext (
            fc2GuiContext context )
```

Destroy a GUI context.

Parameters

context	Context to be destroyed.

Returns

An Error indicating the success or failure of the function.

8.3.1.3 fc2Disonnect()

```
FLYCAPTURE2_C_API void fc2Disonnect (
             fc2GuiContext context )
```

Disconnect GUI context from camera.

Parameters

context	GUI context to disconnect.

Returns

An Error indicating the success or failure of the function.

Deprecated This method is deprecated and will be removed in a future FlyCapture2 release. Please use fc2GU↩ IDisconnect instead.

8.3.1.4 fc2GUIConnect()

```
FLYCAPTURE2_C_API void fc2GUIConnect (
             fc2GuiContext context,
             fc2Context cameraContext )
```

Connect GUI context to a camera context.

Parameters

context	GUI context to connect.
cameraContext	Camera context to connect.

Returns

An Error indicating the success or failure of the function.

8.3.1.5 fc2GUIDisconnect()

```
FLYCAPTURE2_C_API void fc2GUIDisconnect (
            fc2GuiContext context )
```

Disconnect GUI context from camera.

Parameters

context	GUI context to disconnect.

Returns

An Error indicating the success or failure of the function.

8.3.1.6 fc2Hide()

```
FLYCAPTURE2_C_API void fc2Hide (
            fc2GuiContext context )
```

Hide the GUI.

Parameters

context	Pointer to context to hide.

Returns

An Error indicating the success or failure of the function.

8.3.1.7 fc2IsVisible()

```
FLYCAPTURE2_C_API BOOL fc2IsVisible (
            fc2GuiContext context )
```

Check if the GUI is visible.

Parameters

context	Pointer to context to show.

Returns

Whether the GUI is visible.

8.3.1.8 fc2Show()

```
FLYCAPTURE2_C_API void fc2Show (
             fc2GuiContext context )
```

Show the GUI.

Parameters

context	Pointer to context to show.

Returns

An Error indicating the success or failure of the function.

8.3.1.9 fc2ShowModal()

```
FLYCAPTURE2_C_API void fc2ShowModal (
             fc2GuiContext context,
             BOOL * pOkSelected,
             fc2PGRGuid * guidArray,
             unsigned int * size )
```

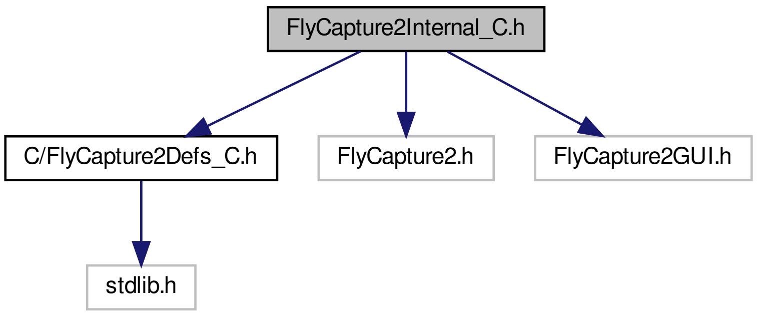
Show the camera selection dialog.

Parameters

context	Pointer to context to show.
pOkSelected	Whether Ok (true) or Cancel (false) was clicked.
guidArray	Array of PGRGuids containing the selected cameras.
size	Size of PGRGuid array.

8.4 FlyCapture2Internal_C.h File Reference

Include dependency graph for FlyCapture2Internal_C.h:



Data Structures

- struct fc2InternalContext
- struct fc2InternalGuiContext
- struct fc2InternalImageCallback

Functions

- bool IsContextValid (fc2Context context)
- bool IsGuiContextValid (fc2GuiContext context)
- void SyncCppImageToStruct (fc2Image ∗pImage)

8.4.1 Function Documentation

8.4.1.1 IsContextValid()

```
bool IsContextValid (
            fc2Context context )  [inline]
```

8.4.1.2 IsGuiContextValid()

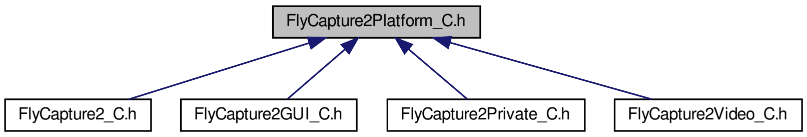
```
bool IsGuiContextValid (
            fc2GuiContext context )  [inline]
```

8.4.1.3 SyncCppImageToStruct()

```
void SyncCppImageToStruct (
            fc2Image * pImage )  [inline]
```

8.5 FlyCapture2Platform_C.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define FLYCAPTURE2_C_API
- #define FLYCAPTURE2_C_CALL_CONVEN

8.5.1 Macro Definition Documentation

8.5.1.1 FLYCAPTURE2_C_API

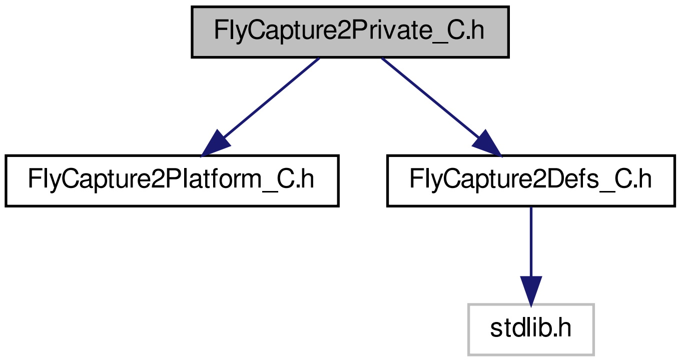
```
#define FLYCAPTURE2_C_API
```

8.5.1.2 FLYCAPTURE2_C_CALL_CONVEN

```
#define FLYCAPTURE2_C_CALL_CONVEN
```

8.6 FlyCapture2Private_C.h File Reference

Include dependency graph for FlyCapture2Private_C.h:



Functions

- FLYCAPTURE2_C_API void ∗ GetInternal (unsigned int index)

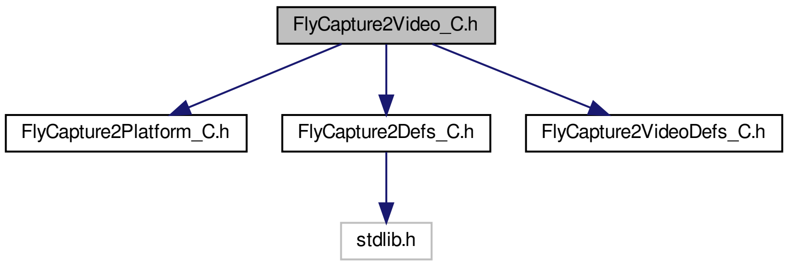
8.6.1 Function Documentation

8.6.1.1 GetInternal()

```
FLYCAPTURE2_C_API void* GetInternal (
             unsigned int index )
```

8.7 FlyCapture2Video_C.h File Reference

Include dependency graph for FlyCapture2Video_C.h:



Functions

- FLYCAPTURE2_C_API fc2Error fc2VideoCreate (fc2VideoContext ∗pVideoContext)

  Create a Video context.
- FLYCAPTURE2_C_API fc2Error fc2VideoAVIOpen (fc2VideoContext VideoContext, const char ∗pFileName, fc2AVIOption ∗pOption)

  Open an AVI file in preparation for writing Images to disk.
- FLYCAPTURE2_C_API fc2Error fc2VideoMJPGOpen (fc2VideoContext VideoContext, const char ∗pFile↩ Name, fc2MJPGOption ∗pOption)

  Open an MJPEG file in preparation for writing Images to disk.
- FLYCAPTURE2_C_API fc2Error fc2VideoH264Open (fc2VideoContext VideoContext, const char ∗pFile↩ Name, fc2H264Option ∗pOption)

  Open an H.264 video file in preparation for writing Images to disk.
- FLYCAPTURE2_C_API fc2Error fc2VideoAppend (fc2VideoContext VideoContext, fc2Image ∗pImage)

  Append an image to the video file.
- FLYCAPTURE2_C_API fc2Error fc2VideoSetMaximumSize (fc2VideoContext VideoContext, unsigned int size)

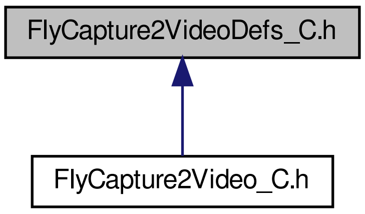
  Set the maximum file size (in megabytes) of a AVI/MP4 file.
- FLYCAPTURE2_C_API fc2Error fc2VideoClose (fc2VideoContext VideoContext)

  Close the video file.
- FLYCAPTURE2_C_API fc2Error fc2VideoDestroy (fc2VideoContext VideoContext)

  Destroy a video context.

8.8 FlyCapture2VideoDefs_C.h File Reference

This graph shows which files directly or indirectly include this file:



Data Structures

- struct fc2MJPGOption

  Options for saving MJPG files.
- struct fc2H264Option

  Options for saving H264 files.
- struct fc2AVIOption

  Options for saving AVI files.

Typedefs

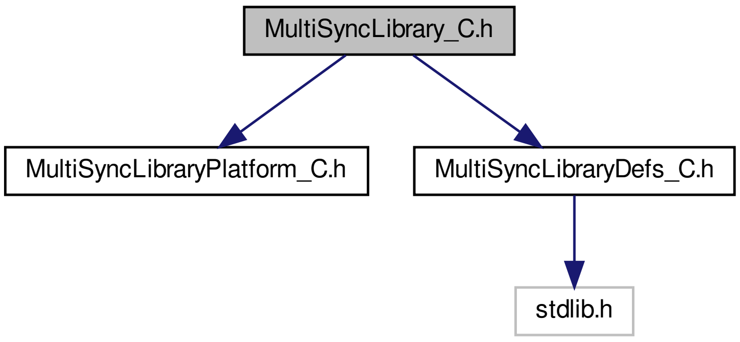
- typedef void ∗ fc2VideoContext

  A context referring to the video recorder object.

8.9 Licensing.dox File Reference

8.10 MultiSyncLibrary_C.h File Reference

Include dependency graph for MultiSyncLibrary_C.h:



Functions

- MULTISYNCLIBRARY_C_API syncError syncCreateContext (syncContext ∗pContext)

  Create a Sync context for MultiSync Library.
- MULTISYNCLIBRARY_C_API syncError syncDestroyContext (syncContext context)

  Destory the sync context.
- MULTISYNCLIBRARY_C_API syncError syncStart (syncContext context)

  Start the sync progress.
- MULTISYNCLIBRARY_C_API syncError syncStop (syncContext context)

  Stop the sync progress.
- MULTISYNCLIBRARY_C_API syncError syncRescanMasterTimingBus (syncContext context)

  Scan newly connected or removed timing bus (for corss-PC syncing only)
- MULTISYNCLIBRARY_C_API syncMessage syncGetStatus (syncContext context)

  Start the sync progress.
- MULTISYNCLIBRARY_C_API double syncGetTimeSinceSynced (syncContext context)

  Time since sync started.
- MULTISYNCLIBRARY_C_API BOOL syncIsTimingBusConnected (syncContext context)

  Whether syncing across PCs.
- MULTISYNCLIBRARY_C_API BOOL syncEnableCrossPCSynchronization (syncContext context)

  Enable across pc synchronization support.
- MULTISYNCLIBRARY_C_API BOOL syncDisableCrossPCSynchronization (syncContext context)

  Disable across pc synchronization support.
- MULTISYNCLIBRARY_C_API BOOL syncQueryCrossPCSynchronizationSetting (syncContext context)

  Query cross pc synchronizaion support status.

8.10.1 Function Documentation

8.10.1.1 syncCreateContext()

```
MULTISYNCLIBRARY_C_API syncError syncCreateContext (
            syncContext * pContext )
```

Create a Sync context for MultiSync Library.

This call must be made before any other calls that use a context will succeed.

Parameters

pContext	A pointer to the syncContext to be created.

Returns

A syncError indicating the success or failure of the function.

8.10.1.2 syncDestroyContext()

```
MULTISYNCLIBRARY_C_API syncError syncDestroyContext (
            syncContext context )
```

Destory the sync context.

This must be called when the user is finished with the context in order to prevent memory leaks.

Parameters

context	The syncContext to be destoryed.

Returns

A syncError indicating the success or failure of the function.

8.10.1.3 syncDisableCrossPCSynchronization()

```
MULTISYNCLIBRARY_C_API BOOL syncDisableCrossPCSynchronization (
            syncContext context )
```

Disable across pc synchronization support.

Parameters

context	The syncContext to be used.

Returns

True if operation was successful

8.10.1.4 syncEnableCrossPCSynchronization()

```
MULTISYNCLIBRARY_C_API BOOL syncEnableCrossPCSynchronization (
            syncContext context )
```

Enable across pc synchronization support.

Parameters

context	The syncContext to be used.

Returns

True if operation was successful

8.10.1.5 syncGetStatus()

```
MULTISYNCLIBRARY_C_API syncMessage syncGetStatus (
            syncContext context )
```

Start the sync progress.

Parameters

context	The syncContext to be used.

Returns

A syncMessage indicating the sync status.

8.10.1.6 syncGetTimeSinceSynced()

```
MULTISYNCLIBRARY_C_API double syncGetTimeSinceSynced (
            syncContext context )
```

Time since sync started.

Parameters

context	The syncContext to be used.

Returns

Time sinced synced.

8.10.1.7 syncIsTimingBusConnected()

```
MULTISYNCLIBRARY_C_API BOOL syncIsTimingBusConnected (
            syncContext context )
```

Whether syncing across PCs.

Parameters

context	The syncContext to be used.

Returns

True if its syncing across PC

8.10.1.8 syncQueryCrossPCSynchronizationSetting()

```
MULTISYNCLIBRARY_C_API BOOL syncQueryCrossPCSynchronizationSetting (
            syncContext context )
```

Query cross pc synchronizaion support status.

Parameters

context	The syncContext to be used.

Returns

True if cross pc synchronization was supported

8.10.1.9 syncRescanMasterTimingBus()

```
MULTISYNCLIBRARY_C_API syncError syncRescanMasterTimingBus (
            syncContext context )
```

Scan newly connected or removed timing bus (for corss-PC syncing only)

Parameters

context	The syncContext to be used.

Returns

A syncError indicating the success or failure of the function.

8.10.1.10 syncStart()

```
MULTISYNCLIBRARY_C_API syncError syncStart (
             syncContext context )
```

Start the sync progress.

Parameters

context	The syncContext to be used.

Returns

A syncError indicating the success or failure of the function.

8.10.1.11 syncStop()

```
MULTISYNCLIBRARY_C_API syncError syncStop (
             syncContext context )
```

Stop the sync progress.

Parameters

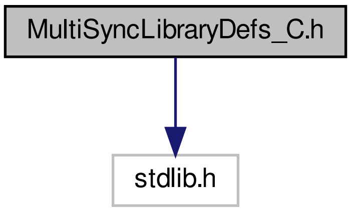
context	The syncContext to be used.

Returns

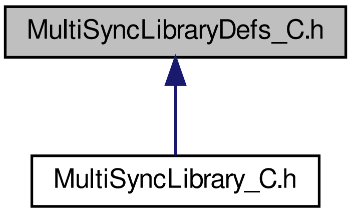
A syncError indicating the success or failure of the function.

8.11 MultiSyncLibraryDefs_C.h File Reference

Include dependency graph for MultiSyncLibraryDefs_C.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define FALSE 0
- #define TRUE 1
- #define FULL_32BIT_VALUE 0x7FFFFFFF
- #define MAX_STRING_LENGTH 512

Typedefs

- typedef int BOOL
- typedef void ∗ syncContext

  A context to the MultiSync C library.

Enumerations

- enum syncError {
  SYNC_ERROR_OK = 0,
  SYNC_ERROR_FAILED,
  SYNC_ERROR_ALREADY_STARTED,
  SYNC_ERROR_ALREADY_STOPPED,
  SYNC_ERROR_CONTEXT_NOT_INITIALIZED,
  SYNC_ERROR_UNKNOWN_ERROR }
- enum syncMessage {
  SYNC_MESSAGE_OK = 0,
  SYNC_MESSAGE_FAILED,
  SYNC_MESSAGE_STARTED,
  SYNC_MESSAGE_STOPPED,
  SYNC_MESSAGE_SYNCING,
  SYNC_MESSAGE_NOMASTER,
  SYNC_MESSAGE_THREAD_ERROR,
  SYNC_MESSAGE_DEVICE_ERROR,
  SYNC_MESSAGE_NOT_ENOUGH_DEVICES,
  SYNC_MESSAGE_BUS_RESET,
  SYNC_MESSAGE_NOT_INITIALIZED,
  SYNC_MESSAGE_UNKNOWN_ERROR }

8.11.1 Macro Definition Documentation

8.11.1.1 FALSE

```
#define FALSE 0
```

8.11.1.2 FULL_32BIT_VALUE

```
#define FULL_32BIT_VALUE 0x7FFFFFFF
```

8.11.1.3 MAX_STRING_LENGTH

```
#define MAX_STRING_LENGTH 512
```

8.11.1.4 TRUE

```
#define TRUE 1
```

8.11.2 Typedef Documentation

8.11.2.1 BOOL

```
typedef int BOOL
```

8.11.2.2 syncContext

```
typedef void* syncContext
```

A context to the MultiSync C library.

It must be created before performing any calls to the library.

8.11.3 Enumeration Type Documentation

8.11.3.1 syncError

```
enum syncError
```

Enumerator

SYNC_ERROR_OK	
SYNC_ERROR_FAILED	
SYNC_ERROR_ALREADY_STARTED	
SYNC_ERROR_ALREADY_STOPPED	
SYNC_ERROR_CONTEXT_NOT_INITIALIZED	
SYNC_ERROR_UNKNOWN_ERROR	

8.11.3.2 syncMessage

```
enum syncMessage
```

Enumerator

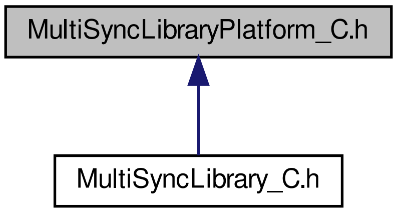
SYNC_MESSAGE_OK	
SYNC_MESSAGE_FAILED	
SYNC_MESSAGE_STARTED	

Enumerator

SYNC_MESSAGE_STOPPED	
SYNC_MESSAGE_SYNCING	
SYNC_MESSAGE_NOMASTER	
SYNC_MESSAGE_THREAD_ERROR	
SYNC_MESSAGE_DEVICE_ERROR	
SYNC_MESSAGE_NOT_ENOUGH_DEVICES	
SYNC_MESSAGE_BUS_RESET	
SYNC_MESSAGE_NOT_INITIALIZED	
SYNC_MESSAGE_UNKNOWN_ERROR	

8.12 MultiSyncLibraryPlatform_C.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define MULTISYNCLIBRARY_C_API
- #define MULTISYNCLIBRARY_C_CALL_CONVEN

8.12.1 Macro Definition Documentation

8.12.1.1 MULTISYNCLIBRARY_C_API

```
#define MULTISYNCLIBRARY_C_API
```

8.12.1.2 MULTISYNCLIBRARY_C_CALL_CONVEN

```
#define MULTISYNCLIBRARY_C_CALL_CONVEN
```

Index

absControl
    fc2TriggerDelay, 214
absMax
    fc2TriggerDelayInfo, 217
absMin
    fc2TriggerDelayInfo, 217
absValSupported
    fc2TriggerDelayInfo, 217
absValue
    fc2TriggerDelay, 214
applicationIPAddress
    fc2CameraInfo, 150
applicationPort
    fc2CameraInfo, 151
asyncBusSpeed
    fc2Config, 160
autoManualMode
    fc2TriggerDelay, 215
autoSupported
    fc2TriggerDelayInfo, 217
available
    fc2EmbeddedImageInfoProperty, 168

BOOL
    MultiSyncLibraryDefs_C.h, 268
    TypeDefs, 116
bandwidthAllocation
    fc2Config, 160
bayerFormat
    fc2Image, 189
bayerTileFormat
    fc2CameraInfo, 151
binaryFile
    fc2PGMOption, 201
    fc2PPMOption, 204
bitrate
    fc2H264Option, 188
brightness
    fc2EmbeddedImageInfo, 167
build
    fc2Version, 224
Bus Manager Operation, 11
    fc2DiscoverGigECameras, 12
    fc2FireBusReset, 13
    fc2ForceAllIPAddressesAutomatically, 13
    fc2ForceIPAddressAutomatically, 13
    fc2ForceIPAddressToCamera, 14
    fc2GetCameraFromIPAddress, 16
    fc2GetCameraFromIndex, 14
    fc2GetCameraFromSerialNumber, 16
    fc2GetCameraSerialNumberFromIndex, 17
    fc2GetDeviceFromIndex, 17
    fc2GetInterfaceTypeFromGuid, 18
    fc2GetNumOfCameras, 18
    fc2GetNumOfDevices, 18
    fc2GetTopology, 19
    fc2GetUsbLinkInfo, 19
    fc2GetUsbPortStatus, 20
    fc2IsCameraControlable, 20
    fc2ReadPhyRegister, 21
    fc2RegisterCallback, 21
    fc2RescanBus, 22
    fc2UnregisterCallback, 22
    fc2WritePhyRegister, 22
busNumber
    fc2CameraInfo, 151
byteOrder
    fc2SystemInfo, 209

cameraCurrents
    fc2CameraStats, 157
cameraPowerUp
    fc2CameraStats, 157
cameraVoltages
    fc2CameraStats, 157
ccpStatus
    fc2CameraInfo, 151
chipIdHi
    fc2ConfigROM, 164
chipIdLo
    fc2ConfigROM, 164
cols
    fc2Image, 189
compression
    fc2TIFFOption, 211
compressionLevel
    fc2PNGOption, 203
configROM
    fc2CameraInfo, 151
Connection and Image Retrieval, 25
    fc2Connect, 25
    fc2Disconnect, 26
    fc2GetConfiguration, 26
    fc2IsConnected, 27
    fc2RetrieveBuffer, 27
    fc2SetCallback, 28
    fc2SetConfiguration, 28
    fc2SetUserBuffers, 29
    fc2StartCapture, 29
    fc2StartCaptureCallback, 30

fc2StartSyncCapture, 30
fc2StartSyncCaptureCallback, 31
fc2StopCapture, 32
fc2WaitForBufferEvent, 32
cpuDescription
fc2SystemInfo, 209
cycleCount
fc2TimeStamp, 212
cycleOffset
fc2TimeStamp, 212
cycleSeconds
fc2TimeStamp, 212

DCAM Formats, 64
fc2GetVideoModeAndFrameRate, 64
fc2GetVideoModeAndFrameRateInfo, 65
fc2SetVideoModeAndFrameRate, 65
dataSize
fc2Image, 189
defaultGateway
fc2CameraInfo, 151
delay
fc2StrobeControl, 205
destinationIpAddress
fc2GigEStreamChannel, 186
doNotFragment
fc2GigEStreamChannel, 186
driverList
fc2SystemInfo, 209
driverName
fc2CameraInfo, 152
driverType
fc2CameraInfo, 152
duration
fc2StrobeControl, 205

embeddedBrightness
fc2ImageMetadata, 191
embeddedExposure
fc2ImageMetadata, 191
embeddedFrameCounter
fc2ImageMetadata, 191
embeddedGPIOPinState
fc2ImageMetadata, 191
embeddedGain
fc2ImageMetadata, 191
embeddedROIPosition
fc2ImageMetadata, 191
embeddedShutter
fc2ImageMetadata, 192
embeddedStrobePattern
fc2ImageMetadata, 192
embeddedTimeStamp
fc2ImageMetadata, 192
embeddedWhiteBalance
fc2ImageMetadata, 192
enablePacketResend
fc2GigEConfig, 179
enabled
fc2LUTData, 198
Enumerations, 118
fc2BandwidthAllocation, 122
fc2BayerTileFormat, 123
fc2BusCallbackType, 123
fc2BusSpeed, 124
fc2ColorProcessingAlgorithm, 124
fc2DriverType, 125
fc2Error, 125
fc2FrameRate, 126
fc2GrabMode, 127
fc2GrabTimeout, 127
fc2ImageFileFormat, 128
fc2InterfaceType, 128
fc2Mode, 128
fc2PCIeBusSpeed, 129
fc2PixelFormat, 130
fc2PropertyType, 130
fc2VideoMode, 131
EventCallbackFcn
fc2EventOptions, 171
EventData
fc2EventCallbackData, 169
EventDataSize
fc2EventCallbackData, 169
EventID
fc2EventCallbackData, 170
EventName
fc2EventCallbackData, 170
fc2EventOptions, 171
EventTimestamp
fc2EventCallbackData, 170
EventUserData
fc2EventCallbackData, 170
fc2EventOptions, 171
EventUserDataSize
fc2EventCallbackData, 170
fc2EventOptions, 171
exposure
fc2EmbeddedImageInfo, 167

FALSE
MultiSyncLibraryDefs_C.h, 267
TypeDefs, 115
FLYCAPTURE2_C_API
FlyCapture2Platform_C.h, 256
FLYCAPTURE2_C_CALL_CONVEN
FlyCapture2Platform_C.h, 256
FULL_32BIT_VALUE
MultiSyncLibraryDefs_C.h, 267
TypeDefs, 115
fc2AVIOption, 147
frameRate, 147
reserved, 147
fc2AsyncCommandCallback
Image saving structures., 139
fc2BMPOption, 148
indexedColor_8bit, 148
reserved, 148

fc2BandwidthAllocation
    Enumerations, 122
fc2BayerTileFormat
    Enumerations, 123
fc2BusCallbackType
    Enumerations, 123
fc2BusEventCallback
    Image saving structures., 139
fc2BusSpeed
    Enumerations, 124
fc2ByteOrder
    FlyCapture2Defs_C.h, 248
fc2CalculateImageStatistics
    Image Operation, 86
fc2CallbackHandle
    Image saving structures., 139
fc2CameraEventCallback
    Image saving structures., 139
fc2CameraInfo, 149
    applicationIPAddress, 150
    applicationPort, 151
    bayerTileFormat, 151
    busNumber, 151
    ccpStatus, 151
    configROM, 151
    defaultGateway, 151
    driverName, 152
    driverType, 152
    firmwareBuildTime, 152
    firmwareVersion, 152
    gigEMajorVersion, 152
    gigEMinorVersion, 152
    iidcVer, 153
    interfaceType, 153
    ipAddress, 153
    isColorCamera, 153
    macAddress, 153
    maximumBusSpeed, 153
    modelName, 154
    nodeNumber, 154
    pcieBusSpeed, 154
    reserved, 154
    sensorInfo, 154
    sensorResolution, 154
    serialNumber, 155
    subnetMask, 155
    userDefinedName, 155
    vendorName, 155
    xmlURL1, 155
    xmlURL2, 155
fc2CameraStats, 156
    cameraCurrents, 157
    cameraPowerUp, 157
    cameraVoltages, 157
    imageCorrupt, 157
    imageDriverDropped, 157
    imageDropped, 157
    imageXmitFailed, 157
    numCurrents, 158
    numResendPacketsReceived, 158
    numResendPacketsRequested, 158
    numVoltages, 158
    portErrors, 158
    regReadFailed, 158
    regWriteFailed, 158
    reserved, 159
    temperature, 159
    timeSinceBusReset, 159
    timeSinceInitialization, 159
    timeStamp, 159
fc2CheckDriver
    Utilities, 110
fc2ColorProcessingAlgorithm
    Enumerations, 124
fc2Config, 159
    asyncBusSpeed, 160
    bandwidthAllocation, 160
    grabMode, 161
    grabTimeout, 161
    highPerformanceRetrieveBuffer, 161
    isochBusSpeed, 161
    minNumImageNotifications, 161
    numBuffers, 162
    numImageNotifications, 162
    registerTimeout, 162
    registerTimeoutRetries, 162
    reserved, 163
fc2ConfigROM, 163
    chipIdHi, 164
    chipIdLo, 164
    nodeVendorId, 164
    pszKeyword, 164
    reserved, 164
    unitSWVer, 165
    unitSpecId, 164
    unitSubSWVer, 165
    vendorUniqueInfo_0, 165
    vendorUniqueInfo_1, 165
    vendorUniqueInfo_2, 165
    vendorUniqueInfo_3, 165
fc2Connect
    Connection and Image Retrieval, 25
fc2Context
    TypeDefs, 116
fc2ConvertImage
    Image Operation, 86
fc2ConvertImageTo
    Image Operation, 87
fc2CreateContext
    FlyCapture2_C.h, 236
fc2CreateGUIContext
    FlyCapture2GUI_C.h, 251
fc2CreateGigEContext
    FlyCapture2_C.h, 237
fc2CreateImage
    Image Operation, 87

fc2CreateImageStatistics
Image Statistics Operation, 96
fc2CreateTopologyNode
TopologyNode Operation, 104
fc2DeregisterAllEvents
FlyCapture2_C.h, 237
fc2DeregisterEvent
FlyCapture2_C.h, 237
fc2DestroyContext
FlyCapture2_C.h, 238
fc2DestroyGUIContext
FlyCapture2GUI_C.h, 251
fc2DestroyImage
Image Operation, 88
fc2DestroyImageStatistics
Image Statistics Operation, 96
fc2DestroyTopologyNode
TopologyNode Operation, 104
fc2DetermineBitsPerPixel
Image Operation, 88
fc2Disconnect
Connection and Image Retrieval, 26
fc2DiscoverGigECameras
Bus Manager Operation, 12
fc2DiscoverGigEPacketSize
GigE property manipulation, 73
fc2Disonnect
FlyCapture2GUI_C.h, 252
fc2DriverType
Enumerations, 125
fc2EmbeddedImageInfo, 166
brightness, 167
exposure, 167
frameCounter, 167
GPIOPinState, 167
gain, 167
ROIPosition, 167
shutter, 167
strobePattern, 168
timestamp, 168
whiteBalance, 168
fc2EmbeddedImageInfoProperty, 168
available, 168
onOff, 169
fc2EnableLUT
Look Up Table, 51
fc2Error
Enumerations, 125
fc2ErrorToDescription
Utilities, 111
fc2EventCallbackData, 169
EventData, 169
EventDataSize, 169
EventID, 170
EventName, 170
EventTimestamp, 170
EventUserData, 170
EventUserDataSize, 170
fc2EventOptions, 171
EventCallbackFcn, 171
EventName, 171
EventUserData, 171
EventUserDataSize, 171
fc2FireBusReset
Bus Manager Operation, 13
fc2FireSoftwareTrigger
Trigger, 41
fc2FireSoftwareTriggerBroadcast
Trigger, 42
fc2ForceAllIPAddressesAutomatically
Bus Manager Operation, 13
fc2ForceIPAddressAutomatically
Bus Manager Operation, 13
fc2ForceIPAddressToCamera
Bus Manager Operation, 14
fc2Format7ImageSettings, 172
height, 172
mode, 172
offsetX, 173
offsetY, 173
pixelFormat, 173
reserved, 173
width, 173
fc2Format7Info, 174
imageHStepSize, 174
imageVStepSize, 174
maxHeight, 175
maxPacketSize, 175
maxWidth, 175
minPacketSize, 175
mode, 175
offsetHStepSize, 175
offsetVStepSize, 176
packetSize, 176
percentage, 176
pixelFormatBitField, 176
reserved, 176
vendorPixelFormatBitField, 176
fc2Format7PacketInfo, 177
maxBytesPerPacket, 177
recommendedBytesPerPacket, 177
reserved, 178
unitBytesPerPacket, 178
fc2FrameRate
Enumerations, 126
fc2GUIConnect
FlyCapture2GUI_C.h, 252
fc2GUIDisconnect
FlyCapture2GUI_C.h, 252
fc2GetActiveLUTBank
Look Up Table, 52
fc2GetCameraFromIPAddress
Bus Manager Operation, 16
fc2GetCameraFromIndex
Bus Manager Operation, 14
fc2GetCameraFromSerialNumber

Bus Manager Operation, 16
fc2GetCameraInfo
Information and Properties, 34
fc2GetCameraSerialNumberFromIndex
Bus Manager Operation, 17
fc2GetChannelHistogram
Image Statistics Operation, 96
fc2GetChannelMean
Image Statistics Operation, 97
fc2GetChannelNumPixelValues
Image Statistics Operation, 97
fc2GetChannelPixelValueRange
Image Statistics Operation, 98
fc2GetChannelRange
Image Statistics Operation, 98
fc2GetChannelStatus
Image Statistics Operation, 99
fc2GetConfiguration
Connection and Image Retrieval, 26
fc2GetCycleTime
FlyCapture2_C.h, 238
fc2GetDefaultColorProcessing
Image Operation, 88
fc2GetDefaultOutputFormat
Image Operation, 89
fc2GetDeviceFromIndex
Bus Manager Operation, 17
fc2GetDriverDeviceName
Utilities, 111
fc2GetEmbeddedImageInfo
Memory Channels, 56
fc2GetFormat7Configuration
Format7, 66
fc2GetFormat7Info
Format7, 67
fc2GetGPIOPinDirection
General Purpose Input / Output, 38
fc2GetGigEConfig
GigE image stream configuration, 82
fc2GetGigEImageBinningSettings
GigE image binning settings, 80
fc2GetGigEImageSettings
GigE image settings, 75
fc2GetGigEImageSettingsInfo
GigE image settings, 76
fc2GetGigEImagingMode
GigE image settings, 76
fc2GetGigEProperty
GigE property manipulation, 73
fc2GetGigEStreamChannelInfo
GigE image stream configuration, 82
fc2GetImageColorProcessing
Image Operation, 89
fc2GetImageData
Image Operation, 90
fc2GetImageDimensions
Image Operation, 90
fc2GetImageMetadata
Image Operation, 91
fc2GetImageStatistics
Image Statistics Operation, 99
fc2GetImageTimeStamp
Image Operation, 91
fc2GetInterfaceTypeFromGuid
Bus Manager Operation, 18
fc2GetLUTBankInfo
Look Up Table, 52
fc2GetLUTChannel
Look Up Table, 53
fc2GetLUTInfo
Look Up Table, 53
fc2GetLibraryVersion
Utilities, 111
fc2GetMemoryChannel
Memory Channels, 57
fc2GetMemoryChannelInfo
Memory Channels, 57
fc2GetNumOfCameras
Bus Manager Operation, 18
fc2GetNumOfDevices
Bus Manager Operation, 18
fc2GetNumStreamChannels
GigE image stream configuration, 83
fc2GetProperty
Information and Properties, 34
fc2GetPropertyInfo
Information and Properties, 35
fc2GetRegisterString
Register Operation, 60
fc2GetStats
FlyCapture2_C.h, 239
fc2GetStrobe
Strobe, 48
fc2GetStrobeInfo
Strobe, 49
fc2GetSystemInfo
Utilities, 112
fc2GetTopology
Bus Manager Operation, 19
fc2GetTriggerDelay
Trigger, 42
fc2GetTriggerDelayInfo
Trigger, 43
fc2GetTriggerMode
Trigger, 43
fc2GetTriggerModeInfo
Trigger, 44
fc2GetUsbLinkInfo
Bus Manager Operation, 19
fc2GetUsbPortStatus
Bus Manager Operation, 20
fc2GetVideoModeAndFrameRate
DCAM Formats, 64
fc2GetVideoModeAndFrameRateInfo
DCAM Formats, 65
fc2GigEConfig, 178

enablePacketResend, 179
registerTimeout, 179
registerTimeoutRetries, 179
reserved, 179
fc2GigEImageSettings, 179
height, 180
offsetX, 180
offsetY, 180
pixelFormat, 180
reserved, 181
width, 181
fc2GigEImageSettingsInfo, 181
imageHStepSize, 182
imageVStepSize, 182
maxHeight, 182
maxWidth, 182
offsetHStepSize, 182
offsetVStepSize, 182
pixelFormatBitField, 183
reserved, 183
vendorPixelFormatBitField, 183
fc2GigEProperty, 183
isReadable, 184
isWritable, 184
max, 184
min, 184
propType, 184
reserved, 184
value, 185
fc2GigEPropertyType
GigE specific enumerations, 133
fc2GigEStreamChannel, 185
destinationIpAddress, 186
doNotFragment, 186
hostPort, 186
interPacketDelay, 186
networkInterfaceIndex, 186
packetSize, 186
reserved, 187
sourcePort, 187
fc2GrabMode
Enumerations, 127
fc2GrabTimeout
Enumerations, 127
fc2GuiContext
TypeDefs, 116
fc2H264Option, 187
bitrate, 188
frameRate, 188
height, 188
reserved, 188
width, 188
fc2Hide
FlyCapture2GUI_C.h, 253
fc2IPAddress, 195
octets, 195
fc2Image, 189
bayerFormat, 189
cols, 189
dataSize, 189
format, 189
imageImpl, 189
pData, 189
receivedDataSize, 190
rows, 190
stride, 190
fc2ImageEventCallback
Image saving structures., 139
fc2ImageFileFormat
Enumerations, 128
fc2ImageImpl
TypeDefs, 116
fc2ImageMetadata, 190
embeddedBrightness, 191
embeddedExposure, 191
embeddedFrameCounter, 191
embeddedGPIOPinState, 191
embeddedGain, 191
embeddedROIPosition, 191
embeddedShutter, 192
embeddedStrobePattern, 192
embeddedTimeStamp, 192
embeddedWhiteBalance, 192
reserved, 192
fc2ImageStatisticsContext
TypeDefs, 116
fc2ImageStatisticsDisableAll
Image Statistics Operation, 100
fc2ImageStatisticsEnableAll
Image Statistics Operation, 100
fc2ImageStatisticsEnableGreyOnly
Image Statistics Operation, 100
fc2ImageStatisticsEnableHSLOnly
Image Statistics Operation, 101
fc2ImageStatisticsEnableRGBOnly
Image Statistics Operation, 101
fc2InterfaceType
Enumerations, 128
fc2InternalContext, 193
pBusMgr, 193
pCamera, 193
fc2InternalGuiContext, 193
pCameraControlDlg, 193
pCameraSelectionDlg, 193
fc2InternalImageCallback, 194
pCallback, 194
pCallbackData, 194
fc2IsCameraControlable
Bus Manager Operation, 20
fc2IsConnected
Connection and Image Retrieval, 27
fc2IsVisible
FlyCapture2GUI_C.h, 253
fc2JPEGOption, 195
progressive, 196
quality, 196

reserved, 196
fc2JPG2Option, 196
quality, 197
reserved, 197
fc2LUTData, 197
enabled, 198
inputBitDepth, 198
numBanks, 198
numChannels, 198
numEntries, 198
outputBitDepth, 198
reserved, 199
supported, 199
fc2LaunchBrowser
Utilities, 112
fc2LaunchCommand
Utilities, 113
fc2LaunchCommandAsync
Utilities, 113
fc2LaunchHelp
Utilities, 113
fc2MACAddress, 199
octets, 199
fc2MJPGOption, 200
frameRate, 200
quality, 200
reserved, 200
fc2Mode
Enumerations, 128
fc2NodeType
FlyCapture2Defs_C.h, 249
fc2OSType
FlyCapture2Defs_C.h, 249
fc2PCIeBusSpeed
Enumerations, 129
fc2PGMOption, 201
binaryFile, 201
reserved, 201
fc2PGRGuid, 202
value, 202
fc2PNGOption, 202
compressionLevel, 203
interlaced, 203
reserved, 203
fc2PPMOption, 203
binaryFile, 204
reserved, 204
fc2PixelFormat
Enumerations, 130
fc2PortType
FlyCapture2Defs_C.h, 249
fc2PropertyType
Enumerations, 130
fc2QueryGigEImagingMode
GigE image settings, 76
fc2ReadGVCPMemory
GVCP Register Operation, 69
fc2ReadGVCPRegister
GVCP Register Operation, 70
fc2ReadGVCPRegisterBlock
GVCP Register Operation, 70
fc2ReadPhyRegister
Bus Manager Operation, 21
fc2ReadRegister
Register Operation, 60
fc2ReadRegisterBlock
Register Operation, 61
fc2RegisterAllEvents
FlyCapture2_C.h, 239
fc2RegisterCallback
Bus Manager Operation, 21
fc2RegisterEvent
FlyCapture2_C.h, 239
fc2RescanBus
Bus Manager Operation, 22
fc2RestoreFromMemoryChannel
Memory Channels, 58
fc2RetrieveBuffer
Connection and Image Retrieval, 27
fc2SaveImage
Image Operation, 91
fc2SaveImageWithOption
Image Operation, 92
fc2SaveToMemoryChannel
Memory Channels, 58
fc2SetActiveLUTBank
Look Up Table, 54
fc2SetCallback
Connection and Image Retrieval, 28
fc2SetChannelStatus
Image Statistics Operation, 102
fc2SetConfiguration
Connection and Image Retrieval, 28
fc2SetDefaultColorProcessing
Image Operation, 92
fc2SetDefaultOutputFormat
Image Operation, 93
fc2SetEmbeddedImageInfo
Memory Channels, 59
fc2SetFormat7Configuration
Format7, 67
fc2SetFormat7ConfigurationPacket
Format7, 67
fc2SetGPIOPinDirection
General Purpose Input / Output, 39
fc2SetGPIOPinDirectionBroadcast
General Purpose Input / Output, 39
fc2SetGigEConfig
GigE image stream configuration, 83
fc2SetGigEImageBinningSettings
GigE image binning settings, 80
fc2SetGigEImageSettings
GigE image settings, 78
fc2SetGigEImagingMode
GigE image settings, 78
fc2SetGigEProperty

GigE property manipulation, 74
fc2SetGigEStreamChannelInfo
GigE image stream configuration, 84
fc2SetImageColorProcessing
Image Operation, 93
fc2SetImageData
Image Operation, 93
fc2SetImageDimensions
Image Operation, 94
fc2SetLUTChannel
Look Up Table, 54
fc2SetProperty
Information and Properties, 36
fc2SetPropertyBroadcast
Information and Properties, 36
fc2SetStrobe
Strobe, 49
fc2SetStrobeBroadcast
Strobe, 50
fc2SetTriggerDelay
Trigger, 44
fc2SetTriggerDelayBroadcast
Trigger, 45
fc2SetTriggerMode
Trigger, 45
fc2SetTriggerModeBroadcast
Trigger, 46
fc2SetUserBuffers
Connection and Image Retrieval, 29
fc2SetVideoModeAndFrameRate
DCAM Formats, 65
fc2Show
FlyCapture2GUI_C.h, 253
fc2ShowModal
FlyCapture2GUI_C.h, 254
fc2StartCapture
Connection and Image Retrieval, 29
fc2StartCaptureCallback
Connection and Image Retrieval, 30
fc2StartSyncCapture
Connection and Image Retrieval, 30
fc2StartSyncCaptureCallback
Connection and Image Retrieval, 31
fc2StatisticsChannel
FlyCapture2Defs_C.h, 250
fc2StopCapture
Connection and Image Retrieval, 32
fc2StrobeControl, 204
delay, 205
duration, 205
onOff, 205
polarity, 205
reserved, 205
source, 205
fc2StrobeInfo, 206
maxValue, 206
minValue, 206
onOffSupported, 207
polaritySupported, 207
present, 207
readOutSupported, 207
reserved, 207
source, 207
fc2SystemInfo, 208
byteOrder, 209
cpuDescription, 209
driverList, 209
gpuDescription, 209
libraryList, 209
numCpuCores, 209
osDescription, 210
osType, 210
reserved, 210
screenHeight, 210
screenWidth, 210
sysMemSize, 210
fc2TIFFCompressionMethod
Image saving structures., 139
fc2TIFFOption, 211
compression, 211
reserved, 211
fc2TimeStamp, 212
cycleCount, 212
cycleOffset, 212
cycleSeconds, 212
microSeconds, 213
reserved, 213
seconds, 213
fc2TopologyNodeAddChild
TopologyNode Operation, 104
fc2TopologyNodeAddPortType
TopologyNode Operation, 105
fc2TopologyNodeAssignGuidToNode
TopologyNode Operation, 105
fc2TopologyNodeAssignGuidToNodeEx
TopologyNode Operation, 106
fc2TopologyNodeContext
TypeDefs, 117
fc2TopologyNodeGetChild
TopologyNode Operation, 106
fc2TopologyNodeGetDeviceId
TopologyNode Operation, 106
fc2TopologyNodeGetGuid
TopologyNode Operation, 107
fc2TopologyNodeGetInterfaceType
TopologyNode Operation, 107
fc2TopologyNodeGetNodeType
TopologyNode Operation, 108
fc2TopologyNodeGetNumChildren
TopologyNode Operation, 108
fc2TopologyNodeGetNumPorts
TopologyNode Operation, 108
fc2TopologyNodeGetPortType
TopologyNode Operation, 109
fc2TriggerDelay, 213
absControl, 214

absValue, 214
autoManualMode, 215
onOff, 215
onePush, 215
present, 215
reserved, 215
type, 215
valueA, 216
valueB, 216
fc2TriggerDelayInfo, 216
absMax, 217
absMin, 217
absValSupported, 217
autoSupported, 217
manualSupported, 218
max, 218
min, 218
onOffSupported, 218
onePushSupported, 218
pUnitAbbr, 219
pUnits, 219
present, 218
readOutSupported, 219
reserved, 219
type, 219
fc2TriggerMode, 220
mode, 220
onOff, 220
parameter, 220
polarity, 221
reserved, 221
source, 221
fc2TriggerModeInfo, 221
modeMask, 222
onOffSupported, 222
polaritySupported, 222
present, 223
readOutSupported, 223
reserved, 223
softwareTriggerSupported, 223
sourceMask, 223
valueReadable, 223
fc2UnregisterCallback
Bus Manager Operation, 22
fc2ValidateFormat7Settings
Format7, 68
fc2Version, 224
build, 224
major, 224
minor, 225
type, 225
fc2VideoAVIOpen
Video Recording Operation, 142
fc2VideoAppend
Video Recording Operation, 141
fc2VideoClose
Video Recording Operation, 142
fc2VideoContext
TypeDefs, 117
fc2VideoCreate
Video Recording Operation, 142
fc2VideoDestroy
Video Recording Operation, 143
fc2VideoH264Open
Video Recording Operation, 143
fc2VideoMJPGOpen
Video Recording Operation, 144
fc2VideoMode
Enumerations, 131
fc2VideoSetMaximumSize
Video Recording Operation, 144
fc2WaitForBufferEvent
Connection and Image Retrieval, 32
fc2WriteGVCPMemory
GVCP Register Operation, 71
fc2WriteGVCPRegister
GVCP Register Operation, 71
fc2WriteGVCPRegisterBlock
GVCP Register Operation, 71
fc2WriteGVCPRegisterBroadcast
GVCP Register Operation, 72
fc2WritePhyRegister
Bus Manager Operation, 22
fc2WriteRegister
Register Operation, 62
fc2WriteRegisterBlock
Register Operation, 62
fc2WriteRegisterBroadcast
Register Operation, 63
firmwareBuildTime
fc2CameraInfo, 152
firmwareVersion
fc2CameraInfo, 152
FlyCapture2_C.h, 227
fc2CreateContext, 236
fc2CreateGigEContext, 237
fc2DeregisterAllEvents, 237
fc2DeregisterEvent, 237
fc2DestroyContext, 238
fc2GetCycleTime, 238
fc2GetStats, 239
fc2RegisterAllEvents, 239
fc2RegisterEvent, 239
ResetStats, 240
FlyCapture2Defs_C.h, 240
fc2ByteOrder, 248
fc2NodeType, 249
fc2OSType, 249
fc2PortType, 249
fc2StatisticsChannel, 250
FlyCapture2GUI_C.h, 250
fc2CreateGUIContext, 251
fc2DestroyGUIContext, 251
fc2Disonnect, 252
fc2GUIConnect, 252
fc2GUIDisconnect, 252

fc2Hide, 253
fc2IsVisible, 253
fc2Show, 253
fc2ShowModal, 254
FlyCapture2Internal_C.h, 255
IsContextValid, 255
IsGuiContextValid, 255
SyncCppImageToStruct, 255
FlyCapture2Platform_C.h, 256
FLYCAPTURE2_C_API, 256
FLYCAPTURE2_C_CALL_CONVEN, 256
FlyCapture2Private_C.h, 257
GetInternal, 257
FlyCapture2Video_C.h, 257
FlyCapture2VideoDefs_C.h, 258
format
fc2Image, 189
Format7, 66
fc2GetFormat7Configuration, 66
fc2GetFormat7Info, 67
fc2SetFormat7Configuration, 67
fc2SetFormat7ConfigurationPacket, 67
fc2ValidateFormat7Settings, 68
frameCounter
fc2EmbeddedImageInfo, 167
frameRate
fc2AVIOption, 147
fc2H264Option, 188
fc2MJPGOption, 200

GPIOPinState
fc2EmbeddedImageInfo, 167
GVCP Register Operation, 69
fc2ReadGVCPMemory, 69
fc2ReadGVCPRegister, 70
fc2ReadGVCPRegisterBlock, 70
fc2WriteGVCPMemory, 71
fc2WriteGVCPRegister, 71
fc2WriteGVCPRegisterBlock, 71
fc2WriteGVCPRegisterBroadcast, 72
gain
fc2EmbeddedImageInfo, 167
General Purpose Input / Output, 38
fc2GetGPIOPinDirection, 38
fc2SetGPIOPinDirection, 39
fc2SetGPIOPinDirectionBroadcast, 39
GetInternal
FlyCapture2Private_C.h, 257
GigE image binning settings, 80
fc2GetGigEImageBinningSettings, 80
fc2SetGigEImageBinningSettings, 80
GigE image settings, 75
fc2GetGigEImageSettings, 75
fc2GetGigEImageSettingsInfo, 76
fc2GetGigEImagingMode, 76
fc2QueryGigEImagingMode, 76
fc2SetGigEImageSettings, 78
fc2SetGigEImagingMode, 78
GigE image stream configuration, 82
fc2GetGigEConfig, 82
fc2GetGigEStreamChannelInfo, 82
fc2GetNumStreamChannels, 83
fc2SetGigEConfig, 83
fc2SetGigEStreamChannelInfo, 84
GigE property manipulation, 73
fc2DiscoverGigEPacketSize, 73
fc2GetGigEProperty, 73
fc2SetGigEProperty, 74
GigE specific enumerations, 133
fc2GigEPropertyType, 133
GigE specific structures, 136
gigEMajorVersion
fc2CameraInfo, 152
gigEMinorVersion
fc2CameraInfo, 152
gpuDescription
fc2SystemInfo, 209
grabMode
fc2Config, 161
grabTimeout
fc2Config, 161

height
fc2Format7ImageSettings, 172
fc2GigEImageSettings, 180
fc2H264Option, 188
highPerformanceRetrieveBuffer
fc2Config, 161
hostPort
fc2GigEStreamChannel, 186

IIDC specific structures, 137
iidcVer
fc2CameraInfo, 153
Image Operation, 85
fc2CalculateImageStatistics, 86
fc2ConvertImage, 86
fc2ConvertImageTo, 87
fc2CreateImage, 87
fc2DestroyImage, 88
fc2DetermineBitsPerPixel, 88
fc2GetDefaultColorProcessing, 88
fc2GetDefaultOutputFormat, 89
fc2GetImageColorProcessing, 89
fc2GetImageData, 90
fc2GetImageDimensions, 90
fc2GetImageMetadata, 91
fc2GetImageTimeStamp, 91
fc2SaveImage, 91
fc2SaveImageWithOption, 92
fc2SetDefaultColorProcessing, 92
fc2SetDefaultOutputFormat, 93
fc2SetImageColorProcessing, 93
fc2SetImageData, 93
fc2SetImageDimensions, 94
Image saving structures., 138
fc2AsyncCommandCallback, 139
fc2BusEventCallback, 139

fc2CallbackHandle, 139
fc2CameraEventCallback, 139
fc2ImageEventCallback, 139
fc2TIFFCompressionMethod, 139
Image Statistics Operation, 95
fc2CreateImageStatistics, 96
fc2DestroyImageStatistics, 96
fc2GetChannelHistogram, 96
fc2GetChannelMean, 97
fc2GetChannelNumPixelValues, 97
fc2GetChannelPixelValueRange, 98
fc2GetChannelRange, 98
fc2GetChannelStatus, 99
fc2GetImageStatistics, 99
fc2ImageStatisticsDisableAll, 100
fc2ImageStatisticsEnableAll, 100
fc2ImageStatisticsEnableGreyOnly, 100
fc2ImageStatisticsEnableHSLOnly, 101
fc2ImageStatisticsEnableRGBOnly, 101
fc2SetChannelStatus, 102
imageCorrupt
fc2CameraStats, 157
imageDriverDropped
fc2CameraStats, 157
imageDropped
fc2CameraStats, 157
imageHStepSize
fc2Format7Info, 174
fc2GigEImageSettingsInfo, 182
imageImpl
fc2Image, 189
imageVStepSize
fc2Format7Info, 174
fc2GigEImageSettingsInfo, 182
imageXmitFailed
fc2CameraStats, 157
indexedColor_8bit
fc2BMPOption, 148
Information and Properties, 34
fc2GetCameraInfo, 34
fc2GetProperty, 34
fc2GetPropertyInfo, 35
fc2SetProperty, 36
fc2SetPropertyBroadcast, 36
inputBitDepth
fc2LUTData, 198
interPacketDelay
fc2GigEStreamChannel, 186
interfaceType
fc2CameraInfo, 153
interlaced
fc2PNGOption, 203
ipAddress
fc2CameraInfo, 153
isColorCamera
fc2CameraInfo, 153
IsContextValid
FlyCapture2Internal_C.h, 255
IsGuiContextValid
FlyCapture2Internal_C.h, 255
isReadable
fc2GigEProperty, 184
isWritable
fc2GigEProperty, 184
isochBusSpeed
fc2Config, 161

libraryList
fc2SystemInfo, 209
Licensing.dox, 259
Look Up Table, 51
fc2EnableLUT, 51
fc2GetActiveLUTBank, 52
fc2GetLUTBankInfo, 52
fc2GetLUTChannel, 53
fc2GetLUTInfo, 53
fc2SetActiveLUTBank, 54
fc2SetLUTChannel, 54

MAX_STRING_LENGTH
MultiSyncLibraryDefs_C.h, 267
TypeDefs, 115
MULTISYNCLIBRARY_C_API
MultiSyncLibraryPlatform_C.h, 269
MULTISYNCLIBRARY_C_CALL_CONVEN
MultiSyncLibraryPlatform_C.h, 269
macAddress
fc2CameraInfo, 153
major
fc2Version, 224
manualSupported
fc2TriggerDelayInfo, 218
max
fc2GigEProperty, 184
fc2TriggerDelayInfo, 218
maxBytesPerPacket
fc2Format7PacketInfo, 177
maxHeight
fc2Format7Info, 175
fc2GigEImageSettingsInfo, 182
maxPacketSize
fc2Format7Info, 175
maxValue
fc2StrobeInfo, 206
maxWidth
fc2Format7Info, 175
fc2GigEImageSettingsInfo, 182
maximumBusSpeed
fc2CameraInfo, 153
Memory Channels, 56
fc2GetEmbeddedImageInfo, 56
fc2GetMemoryChannel, 57
fc2GetMemoryChannelInfo, 57
fc2RestoreFromMemoryChannel, 58
fc2SaveToMemoryChannel, 58
fc2SetEmbeddedImageInfo, 59
microSeconds

fc2TimeStamp, 213
min
fc2GigEProperty, 184
fc2TriggerDelayInfo, 218
minNumImageNotifications
fc2Config, 161
minPacketSize
fc2Format7Info, 175
minValue
fc2StrobeInfo, 206
minor
fc2Version, 225
mode
fc2Format7ImageSettings, 172
fc2Format7Info, 175
fc2TriggerMode, 220
modeMask
fc2TriggerModeInfo, 222
modelName
fc2CameraInfo, 154
MultiSyncLibrary_C.h, 259
syncCreateContext, 260
syncDestroyContext, 260
syncDisableCrossPCSynchronization, 260
syncEnableCrossPCSynchronization, 261
syncGetStatus, 261
syncGetTimeSinceSynced, 261
syncIsTimingBusConnected, 263
syncQueryCrossPCSynchronizationSetting, 263
syncRescanMasterTimingBus, 263
syncStart, 265
syncStop, 265
MultiSyncLibraryDefs_C.h, 266
BOOL, 268
FALSE, 267
FULL_32BIT_VALUE, 267
MAX_STRING_LENGTH, 267
syncContext, 268
syncError, 268
syncMessage, 268
TRUE, 267
MultiSyncLibraryPlatform_C.h, 269
MULTISYNCLIBRARY_C_API, 269
MULTISYNCLIBRARY_C_CALL_CONVEN, 269

networkInterfaceIndex
fc2GigEStreamChannel, 186
nodeNumber
fc2CameraInfo, 154
nodeVendorId
fc2ConfigROM, 164
numBanks
fc2LUTData, 198
numBuffers
fc2Config, 162
numChannels
fc2LUTData, 198
numCpuCores
fc2SystemInfo, 209
numCurrents
fc2CameraStats, 158
numEntries
fc2LUTData, 198
numImageNotifications
fc2Config, 162
numResendPacketsReceived
fc2CameraStats, 158
numResendPacketsRequested
fc2CameraStats, 158
numVoltages
fc2CameraStats, 158

octets
fc2IPAddress, 195
fc2MACAddress, 199
offsetHStepSize
fc2Format7Info, 175
fc2GigEImageSettingsInfo, 182
offsetVStepSize
fc2Format7Info, 176
fc2GigEImageSettingsInfo, 182
offsetX
fc2Format7ImageSettings, 173
fc2GigEImageSettings, 180
offsetY
fc2Format7ImageSettings, 173
fc2GigEImageSettings, 180
onOff
fc2EmbeddedImageInfoProperty, 169
fc2StrobeControl, 205
fc2TriggerDelay, 215
fc2TriggerMode, 220
onOffSupported
fc2StrobeInfo, 207
fc2TriggerDelayInfo, 218
fc2TriggerModeInfo, 222
onePush
fc2TriggerDelay, 215
onePushSupported
fc2TriggerDelayInfo, 218
osDescription
fc2SystemInfo, 210
osType
fc2SystemInfo, 210
outputBitDepth
fc2LUTData, 198

pBusMgr
fc2InternalContext, 193
pCallback
fc2InternalImageCallback, 194
pCallbackData
fc2InternalImageCallback, 194
pCamera
fc2InternalContext, 193
pCameraControlDlg
fc2InternalGuiContext, 193
pCameraSelectionDlg

fc2InternalGuiContext, 193
pData
fc2Image, 189
pUnitAbbr
fc2TriggerDelayInfo, 219
pUnits
fc2TriggerDelayInfo, 219
packetSize
fc2Format7Info, 176
fc2GigEStreamChannel, 186
parameter
fc2TriggerMode, 220
pcieBusSpeed
fc2CameraInfo, 154
percentage
fc2Format7Info, 176
pixelFormat
fc2Format7ImageSettings, 173
fc2GigEImageSettings, 180
pixelFormatBitField
fc2Format7Info, 176
fc2GigEImageSettingsInfo, 183
polarity
fc2StrobeControl, 205
fc2TriggerMode, 221
polaritySupported
fc2StrobeInfo, 207
fc2TriggerModeInfo, 222
portErrors
fc2CameraStats, 158
present
fc2StrobeInfo, 207
fc2TriggerDelay, 215
fc2TriggerDelayInfo, 218
fc2TriggerModeInfo, 223
progressive
fc2JPEGOption, 196
propType
fc2GigEProperty, 184
pszKeyword
fc2ConfigROM, 164

quality
fc2JPEGOption, 196
fc2JPG2Option, 197
fc2MJPGOption, 200

ROIPosition
fc2EmbeddedImageInfo, 167
readOutSupported
fc2StrobeInfo, 207
fc2TriggerDelayInfo, 219
fc2TriggerModeInfo, 223
receivedDataSize
fc2Image, 190
recommendedBytesPerPacket
fc2Format7PacketInfo, 177
regReadFailed
fc2CameraStats, 158
regWriteFailed
fc2CameraStats, 158
Register Operation, 60
fc2GetRegisterString, 60
fc2ReadRegister, 60
fc2ReadRegisterBlock, 61
fc2WriteRegister, 62
fc2WriteRegisterBlock, 62
fc2WriteRegisterBroadcast, 63
registerTimeout
fc2Config, 162
fc2GigEConfig, 179
registerTimeoutRetries
fc2Config, 162
fc2GigEConfig, 179
reserved
fc2AVIOption, 147
fc2BMPOption, 148
fc2CameraInfo, 154
fc2CameraStats, 159
fc2Config, 163
fc2ConfigROM, 164
fc2Format7ImageSettings, 173
fc2Format7Info, 176
fc2Format7PacketInfo, 178
fc2GigEConfig, 179
fc2GigEImageSettings, 181
fc2GigEImageSettingsInfo, 183
fc2GigEProperty, 184
fc2GigEStreamChannel, 187
fc2H264Option, 188
fc2ImageMetadata, 192
fc2JPEGOption, 196
fc2JPG2Option, 197
fc2LUTData, 199
fc2MJPGOption, 200
fc2PGMOption, 201
fc2PNGOption, 203
fc2PPMOption, 204
fc2StrobeControl, 205
fc2StrobeInfo, 207
fc2SystemInfo, 210
fc2TIFFOption, 211
fc2TimeStamp, 213
fc2TriggerDelay, 215
fc2TriggerDelayInfo, 219
fc2TriggerMode, 221
fc2TriggerModeInfo, 223
ResetStats
FlyCapture2_C.h, 240
rows
fc2Image, 190

screenHeight
fc2SystemInfo, 210
screenWidth
fc2SystemInfo, 210
seconds
fc2TimeStamp, 213

sensorInfo
fc2CameraInfo, 154
sensorResolution
fc2CameraInfo, 154
serialNumber
fc2CameraInfo, 155
shutter
fc2EmbeddedImageInfo, 167
softwareTriggerSupported
fc2TriggerModeInfo, 223
source
fc2StrobeControl, 205
fc2StrobeInfo, 207
fc2TriggerMode, 221
sourceMask
fc2TriggerModeInfo, 223
sourcePort
fc2GigEStreamChannel, 187
stride
fc2Image, 190
Strobe, 48
fc2GetStrobe, 48
fc2GetStrobeInfo, 49
fc2SetStrobe, 49
fc2SetStrobeBroadcast, 50
strobePattern
fc2EmbeddedImageInfo, 168
Structures, 134
subnetMask
fc2CameraInfo, 155
supported
fc2LUTData, 199
syncContext
MultiSyncLibraryDefs_C.h, 268
SyncCppImageToStruct
FlyCapture2Internal_C.h, 255
syncCreateContext
MultiSyncLibrary_C.h, 260
syncDestroyContext
MultiSyncLibrary_C.h, 260
syncDisableCrossPCSynchronization
MultiSyncLibrary_C.h, 260
syncEnableCrossPCSynchronization
MultiSyncLibrary_C.h, 261
syncError
MultiSyncLibraryDefs_C.h, 268
syncGetStatus
MultiSyncLibrary_C.h, 261
syncGetTimeSinceSynced
MultiSyncLibrary_C.h, 261
syncIsTimingBusConnected
MultiSyncLibrary_C.h, 263
syncMessage
MultiSyncLibraryDefs_C.h, 268
syncQueryCrossPCSynchronizationSetting
MultiSyncLibrary_C.h, 263
syncRescanMasterTimingBus
MultiSyncLibrary_C.h, 263
syncStart
MultiSyncLibrary_C.h, 265
syncStop
MultiSyncLibrary_C.h, 265
sysMemSize
fc2SystemInfo, 210

TRUE
MultiSyncLibraryDefs_C.h, 267
TypeDefs, 116
temperature
fc2CameraStats, 159
timeSinceBusReset
fc2CameraStats, 159
timeSinceInitialization
fc2CameraStats, 159
timeStamp
fc2CameraStats, 159
timestamp
fc2EmbeddedImageInfo, 168
TopologyNode Operation, 103
fc2CreateTopologyNode, 104
fc2DestroyTopologyNode, 104
fc2TopologyNodeAddChild, 104
fc2TopologyNodeAddPortType, 105
fc2TopologyNodeAssignGuidToNode, 105
fc2TopologyNodeAssignGuidToNodeEx, 106
fc2TopologyNodeGetChild, 106
fc2TopologyNodeGetDeviceId, 106
fc2TopologyNodeGetGuid, 107
fc2TopologyNodeGetInterfaceType, 107
fc2TopologyNodeGetNodeType, 108
fc2TopologyNodeGetNumChildren, 108
fc2TopologyNodeGetNumPorts, 108
fc2TopologyNodeGetPortType, 109
Trigger, 41
fc2FireSoftwareTrigger, 41
fc2FireSoftwareTriggerBroadcast, 42
fc2GetTriggerDelay, 42
fc2GetTriggerDelayInfo, 43
fc2GetTriggerMode, 43
fc2GetTriggerModeInfo, 44
fc2SetTriggerDelay, 44
fc2SetTriggerDelayBroadcast, 45
fc2SetTriggerMode, 45
fc2SetTriggerModeBroadcast, 46
type
fc2TriggerDelay, 215
fc2TriggerDelayInfo, 219
fc2Version, 225
TypeDefs, 115
BOOL, 116
FALSE, 115
FULL_32BIT_VALUE, 115
fc2Context, 116
fc2GuiContext, 116
fc2ImageImpl, 116
fc2ImageStatisticsContext, 116
fc2TopologyNodeContext, 117

fc2VideoContext, 117
MAX_STRING_LENGTH, 115
TRUE, 116

unitBytesPerPacket
fc2Format7PacketInfo, 178
unitSWVer
fc2ConfigROM, 165
unitSpecId
fc2ConfigROM, 164
unitSubSWVer
fc2ConfigROM, 165
userDefinedName
fc2CameraInfo, 155
Utilities, 110
fc2CheckDriver, 110
fc2ErrorToDescription, 111
fc2GetDriverDeviceName, 111
fc2GetLibraryVersion, 111
fc2GetSystemInfo, 112
fc2LaunchBrowser, 112
fc2LaunchCommand, 113
fc2LaunchCommandAsync, 113
fc2LaunchHelp, 113

value
fc2GigEProperty, 185
fc2PGRGuid, 202
valueReadable
fc2TriggerModeInfo, 223
valueA
fc2TriggerDelay, 216
valueB
fc2TriggerDelay, 216
vendorName
fc2CameraInfo, 155
vendorPixelFormatBitField
fc2Format7Info, 176
fc2GigEImageSettingsInfo, 183
vendorUniqueInfo_0
fc2ConfigROM, 165
vendorUniqueInfo_1
fc2ConfigROM, 165
vendorUniqueInfo_2
fc2ConfigROM, 165
vendorUniqueInfo_3
fc2ConfigROM, 165
Video Recording Operation, 141
fc2VideoAVIOpen, 142
fc2VideoAppend, 141
fc2VideoClose, 142
fc2VideoCreate, 142
fc2VideoDestroy, 143
fc2VideoH264Open, 143
fc2VideoMJPGOpen, 144
fc2VideoSetMaximumSize, 144
Video saving structures., 146

whiteBalance
fc2EmbeddedImageInfo, 168
width
fc2Format7ImageSettings, 173
fc2GigEImageSettings, 181
fc2H264Option, 188

xmlURL1
fc2CameraInfo, 155
xmlURL2
fc2CameraInfo, 155