

Imperx Tools SDK

3.3.0.53

Generated by Doxygen 1.8.13

Contents

1	Main Page	1
2	Module Index	3
2.1	Modules	3
3	Class Index	5
3.1	Class List	5
4	Module Documentation	7
4.1	Imperx Demosaicing SDK Overview	7
4.1.1	Detailed Description	7
4.2	IpxBayer IpxComponent Header	8
4.2.1	Detailed Description	8
4.3	IpxDisplay IpxComponent Header	9
4.3.1	Detailed Description	9
4.4	Display Component Parameters	10
4.4.1	Detailed Description	10
4.5	IpxImage Header	11
4.5.1	Detailed Description	11
4.5.2	Function Documentation	11
4.5.2.1	IpxInitPixelTypeDescr()	11
4.6	IpxImageApi Header	13
4.6.1	Detailed Description	14

4.6.2	Typedef Documentation	14
4.6.2.1	PAllocFunc	14
4.6.2.2	PFreeFunc	14
4.6.3	Function Documentation	15
4.6.3.1	IpxSetMemoryManager()	15
4.6.3.2	IpxAlloc()	16
4.6.3.3	IpxFree()	16
4.6.3.4	IpxCreateEmptyImageHeader()	17
4.6.3.5	IpxCreateImageHeader()	18
4.6.3.6	IpxInitImageHeader()	19
4.6.3.7	IpxCreateImageData()	20
4.6.3.8	IpxCreateImage()	20
4.6.3.9	IpxReleaseImageHeader()	21
4.6.3.10	IpxReleaseImage()	22
4.6.3.11	IpxCloneImage()	22
4.6.3.12	IpxCloneImageExt()	23
4.6.3.13	IpxCopyImageHeader()	24
4.6.3.14	IpxCopyImage()	24
4.6.3.15	IpxCopyImageChannelChar()	25
4.6.3.16	IpxCopyImageChannelShort()	26
4.6.3.17	IpxCopyImageChannelInt()	27
4.6.3.18	IpxCopyImageChannelFloat()	27
4.7	Image Converter Reference	29
4.7.1	Detailed Description	29
4.8	IpxImageSerializer IpxComponent Header	30
4.8.1	Detailed Description	30
4.9	Image Unpacker Reference	31
4.9.1	Detailed Description	31

4.10 IpxPixelFormat Header	32
4.10.1 Detailed Description	34
4.10.2 Enumeration Type Documentation	34
4.10.2.1 II_PIXEL_ALIGNMENT	35
4.10.2.2 II_PIXEL_CHROMATICITY	35
4.10.2.3 II_PIXEL_BITS	36
4.10.2.4 II_PIXEL_TYPE_DEFINES	37
4.10.3 Function Documentation	38
4.10.3.1 IpxGetRowSize()	38
4.10.3.2 IpxGetRowSizeUnaligned()	38
4.10.3.3 IpxGetPixelFormatTypesNumber()	39
4.10.3.4 IpxIsPixelFormat()	39
4.10.3.5 IpxIsGroup()	39
4.10.3.6 IpxGetColorModelDescription()	40
4.10.3.7 IpxGetColorModelDescr()	40
4.10.3.8 IpxGetPixelFormat()	41
4.10.3.9 IpxGetColorModelName()	41
4.10.3.10 IpxGetChannelSequence()	42
4.10.3.11 IpxGetChannelsNumber()	42
4.10.3.12 IpxGetChannelsDepth()	43
4.10.3.13 IpxGetStartPosition()	43
4.10.3.14 IpxGetChannelIndex()	43
4.10.3.15 IpxCheckChannelNames()	44
4.10.3.16 IpxConvertChannelStr()	44
4.10.3.17 IpxGetChannelName()	45
4.11 IpxToolBase Header	46
4.11.1 Detailed Description	47
4.12 Error Codes	48

4.12.1 Detailed Description	48
4.13 Component Type IDs	49
4.13.1 Detailed Description	49
4.14 IpxTrueSense IpxComponent Header	50
4.14.1 Detailed Description	51
4.15 IpxUserData Header	52
4.15.1 Detailed Description	52
4.15.2 Enumeration Type Documentation	52
4.15.2.1 IPX_USER_DATA	52
4.16 DeBayer Parameters	53
4.16.1 Detailed Description	53
4.16.2 Macro Definition Documentation	53
4.16.2.1 DEBAYER_ALGO_TYPE	53
4.16.2.2 DEBAYER_NOREALLOCT	54
4.17 DeBayer Algorithms	55
4.17.1 Detailed Description	55
4.17.2 Macro Definition Documentation	55
4.17.2.1 BAYER_SIMPLE	56
4.17.2.2 BAYER_GRADIENT	56
4.17.2.3 BAYER_EA	56
4.17.2.4 BAYER_OPENGL_MHC	56
4.17.2.5 BAYER_OPENGL_MMA	56
4.18 IpxBayer C++ Class	57
4.18.1 Detailed Description	57
4.19 IpxBayer C-Interface Functions	58
4.19.1 Detailed Description	58
4.19.2 Function Documentation	58
4.19.2.1 IpxBayer_CreateComponent()	59

4.19.2.2	IpxBayer_DeleteComponent()	59
4.19.2.3	IpxBayer_GetComponent()	59
4.19.2.4	IpxBayer_ConvertImage()	60
4.19.2.5	IpxBayer_AllocData()	60
4.19.2.6	IpxBayer_ReleaseData()	61
4.20	Pre-initialization Parameters	62
4.20.1	Detailed Description	62
4.20.2	Macro Definition Documentation	63
4.20.2.1	IDP_BACKGROUND	63
4.20.2.2	IDP_INIT_FIT	63
4.20.2.3	IDP_INIT_AT_X	64
4.20.2.4	IDP_INIT_AT_Y	64
4.20.2.5	IDP_SMOOTHING	64
4.20.2.6	IDP_OGL_BAYER	65
4.20.2.7	IDP_OGL_TRUESENSE	65
4.20.2.8	IDP_GDI_BAYER	65
4.20.2.9	IDP_GDI_TRUESENSE	66
4.20.2.10	IDP_COMMAND_WINDOW	66
4.20.2.11	IDP_OVERLAY_FONT_DESC_0	66
4.20.2.12	IDP_OVERLAY_FONT_DESC_1	67
4.20.2.13	IDP_OVERLAY_FONT_DESC_2	67
4.20.2.14	IDP_OVERLAY_FONT_DESC_3	67
4.21	Run-time Parameters	68
4.21.1	Detailed Description	68
4.21.2	Macro Definition Documentation	69
4.21.2.1	IDP_SIGNATURE	69
4.21.2.2	IDP_VIEW_FIT	69
4.21.2.3	IDP_VIEW_X	69

4.21.2.4	IDP_VIEW_Y	69
4.21.2.5	IDP_VIEW_SCALE	69
4.21.2.6	IDP_MANAGED_FPS	70
4.21.2.7	IDP_MANAGED_STATE	70
4.21.2.8	IDP_VIEW_CLR	70
4.21.2.9	IDP_VIEW_CURSOR_X	70
4.21.2.10	IDP_VIEW_CURSOR_Y	70
4.21.2.11	IDP_PROC_PROCESSOR	70
4.21.2.12	IDP_PROC_PROCESSOR_TYPE	71
4.21.2.13	IDP_MENU_X	71
4.21.2.14	IDP_MENU_Y	71
4.21.2.15	IDP_MENU_CMD	71
4.22	Software Image Correction Parameters	72
4.22.1	Detailed Description	72
4.22.2	Macro Definition Documentation	73
4.22.2.1	IDP_CORR_MODE	73
4.22.2.2	IDP_CORR_GAIN_R	73
4.22.2.3	IDP_CORR_GAIN_G	73
4.22.2.4	IDP_CORR_GAIN_B	74
4.22.2.5	IDP_CORR_OFFS_R	74
4.22.2.6	IDP_CORR_OFFS_G	74
4.22.2.7	IDP_CORR_OFFS_B	75
4.22.2.8	IDP_CORR_GAMMA	75
4.23	White Balance Correction Parameters	76
4.23.1	Detailed Description	76
4.23.2	Macro Definition Documentation	76
4.23.2.1	IDP_CALC_COEF_R	76
4.23.2.2	IDP_CALC_COEF_G	77

4.23.2.3	IDP_CALC_COEF_B	77
4.24	Overlay Text Parameters	78
4.24.1	Detailed Description	78
4.24.2	Macro Definition Documentation	78
4.24.2.1	IDP_OVERLAY_INDEX	78
4.24.2.2	IDP_OVERLAY_POS	79
4.24.2.3	IDP_OVERLAY_FONT	79
4.24.2.4	IDP_OVERLAY_COLOR	79
4.24.2.5	IDP_OVERLAY_BGMODE	80
4.24.2.6	IDP_OVERLAY_TEXT	80
4.25	Dump Rect Parameters	81
4.25.1	Detailed Description	81
4.25.2	Macro Definition Documentation	81
4.25.2.1	IDP_DUMP_X	81
4.25.2.2	IDP_DUMP_Y	82
4.25.2.3	IDP_DUMP_W	82
4.25.2.4	IDP_DUMP_H	82
4.25.2.5	IDP_DUMP_COLOR	82
4.26	IpxDisplay Command Parameters	83
4.26.1	Detailed Description	83
4.26.2	Macro Definition Documentation	84
4.26.2.1	IDPC_SET_CORRECTION	84
4.26.2.2	IDPC_CMD_VIEW_ZOOM_IN	84
4.26.2.3	IDPC_CMD_VIEW_ZOOM_OUT	85
4.26.2.4	IDPC_CMD_VIEW_ATCENTER	85
4.26.2.5	IDPC_CMD_VIEW_AT	85
4.26.2.6	IDPC_CMD_VIEW_PARAMS	85
4.26.2.7	IDPC_CMD_CORR_CALC	86

4.26.2.8	IDPC_CMD_OVERLAY_SHOW	86
4.26.2.9	IDPC_CMD_OVERLAY_HIDE	86
4.26.2.10	IDPC_CMD_MANAGED_ON	86
4.26.2.11	IDPC_CMD_MANAGED_OFF	87
4.26.2.12	IDPC_CMD_DUMP_ON	87
4.26.2.13	IDPC_CMD_DUMP_OFF	87
4.26.2.14	IDPC_CMD_FILTER_ADD	87
4.26.2.15	IDPC_CMD_FILTER_DEL	88
4.26.2.16	IDPC_CMD_PROC_ADD	88
4.26.2.17	IDPC_CMD_PROC_DEL	88
4.26.2.18	IDPC_CMD_MENU_SHOW	88
4.27	Notifications	89
4.27.1	Detailed Description	89
4.27.2	Macro Definition Documentation	89
4.27.2.1	IPXD_LBUTTON_DOWN	89
4.27.2.2	IPXD_LBUTTON_UP	90
4.27.2.3	IPXD_RBUTTON_DOWN	90
4.27.2.4	IPXD_CURSOR_MOVED	90
4.27.2.5	IPXD_KEY_DOWN	90
4.27.2.6	IPXD_VIEW_CHANGED	90
4.27.2.7	IPXD_CCLR_CHANGED	90
4.27.2.8	IPXD_PLAYBACK_FAILED	90
4.27.2.9	IPXD_ERROR_OPENGL	90
4.28	Translate Flags	91
4.28.1	Detailed Description	91
4.28.2	Macro Definition Documentation	91
4.28.2.1	IDFL_SCR_IMG	91
4.28.2.2	IDFL_IMG_SCR	91

4.29 Fit Modes and Mouse Processing	92
4.29.1 Detailed Description	92
4.29.2 Macro Definition Documentation	92
4.29.2.1 IPXD_FIT_NONE	92
4.29.2.2 IPXD_FIT_WINDOW	92
4.29.2.3 IPXD_FIT_FILL	93
4.29.2.4 IPXD_FIT_FULLSIZE	93
4.29.2.5 IPXD_MOUSE_DEFAULT	93
4.29.2.6 IPXD_MOUSE_SKIP	93
4.29.2.7 IPXD_MOUSE_LOCK	93
4.30 IpxDisplay C++ Class	94
4.30.1 Detailed Description	94
4.31 IpxDisplay C-Interface Functions	95
4.31.1 Detailed Description	95
4.31.2 Function Documentation	95
4.31.2.1 IpxDisplay_CreateComponent()	96
4.31.2.2 IpxDisplay_DeleteComponent()	96
4.31.2.3 IpxDisplay_GetComponent()	97
4.31.2.4 IpxDisplay_Initialize()	98
4.31.2.5 IpxDisplay_DisplayVideo()	98
4.31.2.6 IpxDisplay_DisplayImage()	100
4.31.2.7 IpxDisplay_ConvertImage()	101
4.32 IpxImageConverter C-Interface Functions	102
4.32.1 Detailed Description	102
4.32.2 Function Documentation	102
4.32.2.1 IpxImageConverter_CreateComponent()	103
4.32.2.2 IpxImageConverter_DeleteComponent()	103
4.32.2.3 IpxImageConverter_GetComponent()	103

4.32.2.4	IpxImageConverter_ConvertImage()	104
4.32.2.5	IpxImageConverter_IICConvert()	104
4.33	IpxSerializer Parameters	106
4.33.1	Detailed Description	106
4.33.2	Macro Definition Documentation	107
4.33.2.1	ISP_NO_REALLOC	107
4.33.2.2	ISP_JPEG_QUALITY	107
4.33.2.3	ISP_MIN_QUANTIZER	107
4.33.2.4	ISP_MAX_QUANTIZER	108
4.33.2.5	ISP_TICKS_PER_SEC	108
4.33.2.6	ISP_MOVIE_COMPRESSOR	108
4.33.2.7	ISP_MOVIE_COMPRESSORS	109
4.33.2.8	ISP_ADD_PALETTE	109
4.34	IpxImageSerializer C++ Class	110
4.34.1	Detailed Description	110
4.35	IpxImageSerializer C-Interface Functions	111
4.35.1	Detailed Description	111
4.35.2	Function Documentation	112
4.35.2.1	IpxImageSerializer_CreateComponent()	112
4.35.2.2	IpxImageSerializer_DeleteComponent()	112
4.35.2.3	IpxImageSerializer_GetComponent()	112
4.35.2.4	IpxImageSerializer_StartSeriesRecord()	113
4.35.2.5	IpxImageSerializer_StartMovieRecord()	113
4.35.2.6	IpxImageSerializer_FinishRecord()	114
4.35.2.7	IpxImageSerializer_Save()	114
4.35.2.8	IpxImageSerializer_Load()	115
4.36	IpxImageUnpacker C-Interface Functions	116
4.36.1	Detailed Description	116

4.36.2	Function Documentation	116
4.36.2.1	lpxImageUnpacker_CreateComponent()	116
4.36.2.2	lpxImageUnpacker_DeleteComponent()	117
4.36.2.3	lpxImageUnpacker_GetComponent()	117
4.36.2.4	lpxImageUnpacker_Unpack()	117
4.37	TS CFA Demosaicing algorithm Parameters	119
4.37.1	Detailed Description	119
4.37.2	Macro Definition Documentation	119
4.37.2.1	TS_ALGO_TYPE	119
4.37.2.2	TS_NOREALLOC	120
4.37.2.3	TS_ALGO_NUM	120
4.37.2.4	TSASIMPLEF	120
4.37.2.5	TSASIMPLES	120
4.37.2.6	TSABAYERLIKE	121
4.37.2.7	TSAMEDIUM	121
4.37.2.8	TSAQUALITY	121
4.37.2.9	TRUES_OPENGL_MHC	121
4.37.2.10	TRUES_OPENGL_MMA	121
4.38	TS Misc Parameters	122
4.38.1	Detailed Description	122
4.38.2	Macro Definition Documentation	123
4.38.2.1	TS_THREADS_NUM	123
4.38.2.2	TS_NORM_EN	123
4.38.2.3	TS_HORIZ_MIRRORED	123
4.38.2.4	TS_VER_MIRRORED	124
4.38.2.5	TS_MONO_ENABLED	124
4.38.2.6	TS_IMP_FILTER_ENABLED	124
4.38.2.7	TS_SHARPNESS_ENABLED	125

4.38.2.8 TS_DARKFLOOR	125
4.39 TS Gain Parameters	126
4.39.1 Detailed Description	126
4.39.2 Macro Definition Documentation	127
4.39.2.1 TS_RED_GAIN	127
4.39.2.2 TS_GREEN_GAIN	128
4.39.2.3 TS_BLUE_GAIN	128
4.39.2.4 TS_PAN_GAIN	128
4.39.2.5 TS_GLOBAL_GAIN	129
4.39.2.6 TS_ANALOG_GAIN	129
4.39.2.7 TS_ISO_ANALOGGAIN_0	129
4.39.2.8 TS_ISO_ANALOGGAIN_1	130
4.39.2.9 TS_ISO_ANALOGGAIN_2	130
4.39.2.10 TS_ISO_ANALOGGAIN_3	130
4.39.2.11 TS_ISO_ANALOGGAIN_4	130
4.40 TS ISO Panchromatic Channel Parameters	131
4.40.1 Detailed Description	131
4.40.2 Macro Definition Documentation	132
4.40.2.1 TS_ISO_PANSLOPE_0	132
4.40.2.2 TS_ISO_PANSLOPE_1	132
4.40.2.3 TS_ISO_PANSLOPE_2	133
4.40.2.4 TS_ISO_PANSLOPE_3	133
4.40.2.5 TS_ISO_PANSLOPE_4	133
4.40.2.6 TS_ISO_PANINTERCEPT_0	134
4.40.2.7 TS_ISO_PANINTERCEPT_1	134
4.40.2.8 TS_ISO_PANINTERCEPT_2	134
4.40.2.9 TS_ISO_PANINTERCEPT_3	135
4.40.2.10 TS_ISO_PANINTERCEPT_4	135

4.41 TS ISO Color Slope Parameters	136
4.41.1 Detailed Description	136
4.41.2 Macro Definition Documentation	137
4.41.2.1 TS_ISO_COLORSLOPE_0	137
4.41.2.2 TS_ISO_COLORSLOPE_1	137
4.41.2.3 TS_ISO_COLORSLOPE_2	137
4.41.2.4 TS_ISO_COLORSLOPE_3	138
4.41.2.5 TS_ISO_COLORSLOPE_4	138
4.42 TS ISO Color Intercept Parameters	139
4.42.1 Detailed Description	139
4.42.2 Macro Definition Documentation	140
4.42.2.1 TS_ISO_COLORINTERCEPT_0	140
4.42.2.2 TS_ISO_COLORINTERCEPT_1	140
4.42.2.3 TS_ISO_COLORINTERCEPT_2	140
4.42.2.4 TS_ISO_COLORINTERCEPT_3	141
4.42.2.5 TS_ISO_COLORINTERCEPT_4	141
4.43 TS Sigma Filter Parameters	142
4.43.1 Detailed Description	142
4.43.2 Macro Definition Documentation	143
4.43.2.1 TS_PAN_RADIUS0	143
4.43.2.2 TS_PAN_RADIUS1	144
4.43.2.3 TS_PAN_RADIUS2	144
4.43.2.4 TS_PAN_SIGMA0	144
4.43.2.5 TS_PAN_SIGMA1	145
4.43.2.6 TS_PAN_SIGMA2	145
4.43.2.7 TS_COLOR_RADIUS0	145
4.43.2.8 TS_COLOR_RADIUS1	146
4.43.2.9 TS_COLOR_RADIUS2	146

4.43.2.10 TS_COLOR_SIGMA0	146
4.43.2.11 TS_COLOR_SIGMA1	147
4.43.2.12 TS_COLOR_SIGMA2	147
4.44 TS Coefficients Parameters	148
4.44.1 Detailed Description	148
4.44.2 Macro Definition Documentation	149
4.44.2.1 TS_RR_COEFF	149
4.44.2.2 TS_RG_COEFF	149
4.44.2.3 TS_RB_COEFF	149
4.44.2.4 TS_GR_COEFF	150
4.44.2.5 TS_GG_COEFF	150
4.44.2.6 TS_GB_COEFF	150
4.44.2.7 TS_BR_COEFF	151
4.44.2.8 TS_BG_COEFF	151
4.44.2.9 TS_BB_COEFF	151
4.45 TS Sharpen Parameters	152
4.45.1 Detailed Description	152
4.45.2 Macro Definition Documentation	152
4.45.2.1 TS_SHARPEN_PARAM	152
4.45.2.2 TS_MAX_SHARPEN	153
4.46 TS Noise Threshold Parameters	154
4.46.1 Detailed Description	154
4.46.2 Macro Definition Documentation	154
4.46.2.1 TS_HIGH_LUMA_NOISE	154
4.46.2.2 TS_LOW_LUMA_NOISE	155
4.47 IpxTrueSense C++ Class	156
4.47.1 Detailed Description	156
4.48 IpxTrueSense C-Interface Functions	157
4.48.1 Detailed Description	157
4.48.2 Function Documentation	157
4.48.2.1 IpxTrueSense_CreateComponent()	158
4.48.2.2 IpxTrueSense_DeleteComponent()	158
4.48.2.3 IpxTrueSense_GetComponent()	158
4.48.2.4 IpxTrueSense_ConvertImage()	159
4.48.2.5 IpxTrueSense_AllocData()	159
4.48.2.6 IpxTrueSense_ReleaseData()	160

5	Class Documentation	161
5.1	IpxBayer Class Reference	161
5.1.1	Detailed Description	161
5.1.2	Member Function Documentation	162
5.1.2.1	CreateComponent()	162
5.1.2.2	DeleteComponent()	162
5.1.2.3	GetComponent()	162
5.1.2.4	ConvertImage()	163
5.1.2.5	AllocData()	165
5.1.2.6	ReleaseData()	165
5.2	IpxComponent Class Reference	166
5.2.1	Detailed Description	167
5.2.2	Constructor & Destructor Documentation	167
5.2.2.1	~IpxComponent()	167
5.2.3	Member Function Documentation	167
5.2.3.1	GetComponentTypeID()	167
5.2.3.2	GetParamCount()	167
5.2.3.3	GetParamName()	167
5.2.3.4	GetParamAsString()	168
5.2.3.5	SetParamAsString()	168
5.2.3.6	SetParamBool()	169
5.2.3.7	SetParamInt()	169
5.2.3.8	SetParamFloat()	170
5.2.3.9	SetParamString()	170
5.2.3.10	SetParamArray()	171
5.2.3.11	GetParamBool()	171
5.2.3.12	GetParamInt()	172
5.2.3.13	GetParamFloat()	172

5.2.3.14	GetParamString()	173
5.2.3.15	GetParamArray()	173
5.2.3.16	RunCommand()	174
5.3	IpxDisplay Class Reference	174
5.3.1	Detailed Description	175
5.3.2	Member Function Documentation	175
5.3.2.1	CreateComponent()	175
5.3.2.2	DeleteComponent()	176
5.3.2.3	GetComponent()	176
5.3.2.4	GetSystemInfo()	177
5.3.2.5	Initialize()	177
5.3.2.6	SetVideoMode()	179
5.3.2.7	DisplayVideo()	179
5.3.2.8	DisplayImage()	180
5.3.2.9	ConvertImage()	180
5.3.2.10	Translate()	182
5.4	IpxImage Struct Reference	183
5.4.1	Detailed Description	184
5.4.2	Member Data Documentation	184
5.4.2.1	nSize	184
5.4.2.2	version	185
5.4.2.3	pixelTypeDescr	185
5.4.2.4	origin	185
5.4.2.5	width	185
5.4.2.6	height	185
5.4.2.7	imageSize	185
5.4.2.8	rowSize	185
5.4.2.9	timestamp	186

5.4.2.10	imageID	186
5.4.2.11	userData	186
5.4.2.12	imageData	186
5.4.2.13	imageDataOrigin	186
5.5	IpxImageConverter Class Reference	186
5.5.1	Detailed Description	187
5.5.2	Member Function Documentation	187
5.5.2.1	CreateComponent()	187
5.5.2.2	DeleteComponent()	187
5.5.2.3	GetComponent()	188
5.5.2.4	ConvertImage()	188
5.5.2.5	IIConvert()	189
5.6	IpxImageSerializer Class Reference	189
5.6.1	Detailed Description	190
5.6.2	Member Function Documentation	190
5.6.2.1	CreateComponent()	190
5.6.2.2	DeleteComponent()	190
5.6.2.3	GetComponent()	191
5.6.2.4	StartSeriesRecord()	192
5.6.2.5	StartMovieRecord()	192
5.6.2.6	FinishRecord()	193
5.6.2.7	Save()	193
5.6.2.8	Load()	194
5.6.2.9	GetImageHeader()	194
5.6.2.10	Free()	195
5.7	IpxImageUnpacker Class Reference	195
5.7.1	Detailed Description	196
5.7.2	Member Function Documentation	196

5.7.2.1	CreateComponent()	196
5.7.2.2	DeleteComponent()	196
5.7.2.3	GetComponent()	197
5.7.2.4	Unpack()	197
5.8	IpXImgProcessor Class Reference	198
5.8.1	Detailed Description	198
5.9	IpXPixelTypeDescr Struct Reference	198
5.9.1	Detailed Description	199
5.9.2	Member Data Documentation	199
5.9.2.1	pixelType	199
5.9.2.2	depth	199
5.9.2.3	pixSigned	199
5.9.2.4	pixAlign	199
5.9.2.5	channels	200
5.9.2.6	pixSize	200
5.10	IpXPoint Struct Reference	200
5.10.1	Detailed Description	200
5.10.2	Member Data Documentation	200
5.10.2.1	x	200
5.10.2.2	y	201
5.11	IpXRect Struct Reference	201
5.11.1	Detailed Description	201
5.11.2	Member Data Documentation	201
5.11.2.1	x	201
5.11.2.2	y	201
5.11.2.3	width	202
5.11.2.4	height	202
5.12	IpXSize Struct Reference	202

5.12.1 Detailed Description	202
5.12.2 Member Data Documentation	202
5.12.2.1 width	202
5.12.2.2 height	203
5.13 IpxTrueSense Class Reference	203
5.13.1 Detailed Description	203
5.13.2 Member Function Documentation	204
5.13.2.1 CreateComponent()	204
5.13.2.2 DeleteComponent()	204
5.13.2.3 GetComponent()	204
5.13.2.4 ConvertImage()	205
5.13.2.5 AllocData()	207
5.13.2.6 ReleaseData()	207
5.14 IpxUserData Struct Reference	208
5.14.1 Detailed Description	208
5.14.2 Member Data Documentation	208
5.14.2.1 type	208
5.14.2.2 id	208
5.14.2.3 size	209
5.14.2.4 data	209
5.14.2.5 createdIpx	209
5.14.2.6 pNext	209
Index	211

Chapter 1

Main Page

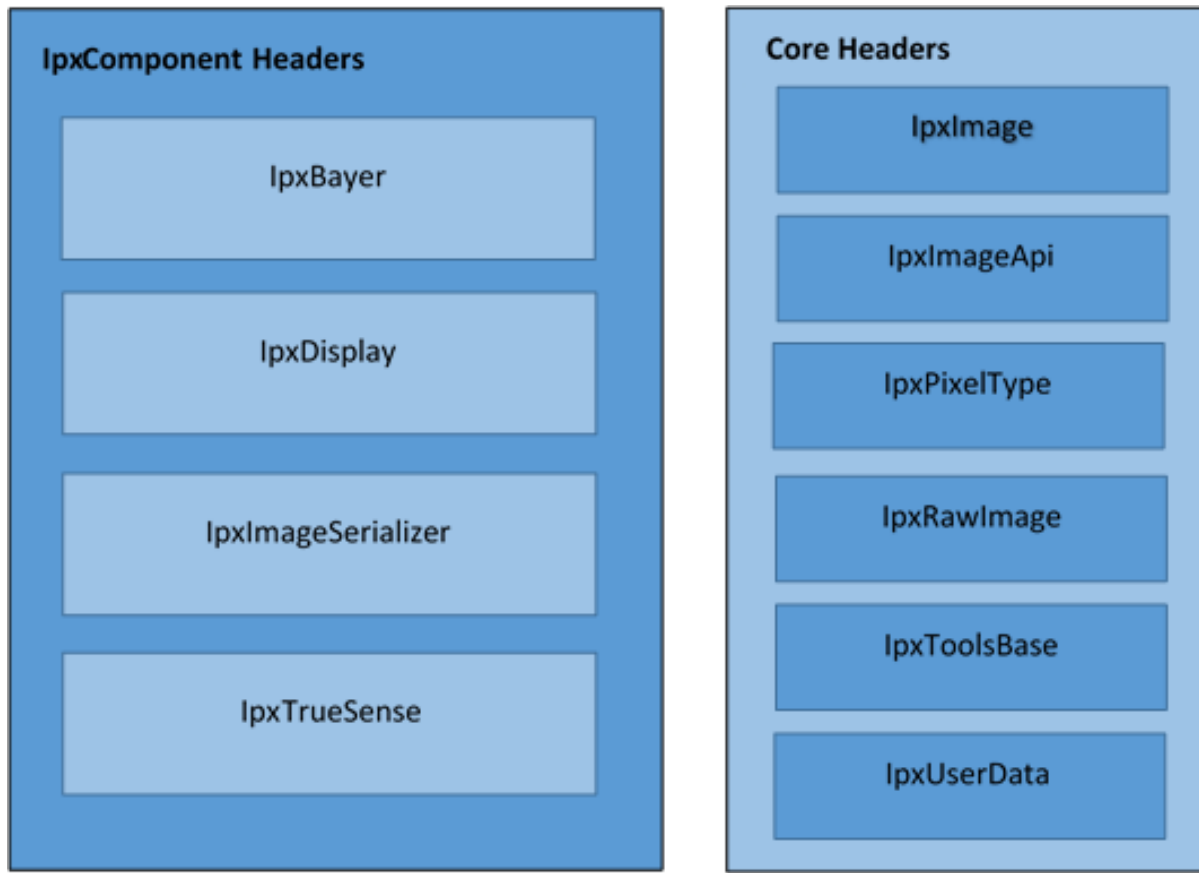
The Imperx Tools is designed to provide software developers with API for ease of integrating Imperx camera's images into their software application. The API includes several component modules implementing the imaging functions.

The API consist of several main classes that implement base [lpxComponent](#) class. The main classes are :

- **[lpxBayer lpxComponent Header](#)** - A header file containing C++ Class and C-Interface functions for [lpxBayer](#) component that contains methods to convert Bayer CFA Demosaic images.
- **[lpxDisplay lpxComponent Header](#)** - A header file containing C++ Class and C-Interface functions for [lpxDisplay](#) component that contains methods to convert and display images.
- **[lpxImageSerializer lpxComponent Header](#)** - A header file containing C++ Class and C-Interface functions for [lpxImageSerializer](#) component that contains methods to serialize [lpxImage](#) images.
- **[lpxTrueSense lpxComponent Header](#)** - A header file containing C++ Class and C-Interface functions for [lpx↔TrueSense](#) component that contains methods to convert TrueSense images.

The Core features consist of defines, macros and functions used for the Imperx Image camera's image manipulation.

- **[lpxImage Header](#)** - A header file containing defines, macros, functions, and data structure for the description of Imperx Images
- **[lpxImageApi Header](#)** - A header file containing Core image functions
- **[lpxPixelFormat Header](#)** - A header file containing the Image Pixel Types
- **[lpxToolBase Header](#)** - A header file containing the defines/macros for errors and the base [lpxComponent](#) class
- **[lpxUserData Header](#)** - A header file containing the user data structure intended to store additional information about the image



IpxTools IpxComponents and Core Headers

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Imperx Demosaicing SDK Overview	7
IpxBayer IpxComponent Header	8
DeBayer Parameters	53
DeBayer Algorithms	55
IpxBayer C++ Class	57
IpxBayer C-Interface Functions	58
IpxTrueSense IpxComponent Header	50
TS CFA Demosaicing algorithm Parameters	119
TS Misc Parameters	122
TS Gain Parameters	126
TS ISO Panchromatic Channel Parameters	131
TS ISO Color Slope Parameters	136
TS ISO Color Intercept Parameters	139
TS Sigma Filter Parameters	142
TS Coefficients Parameters	148
TS Sharpen Parameters	152
TS Noise Threshold Parameters	154
IpxTrueSense C++ Class	156
IpxTrueSense C-Interface Functions	157
IpxDisplay IpxComponent Header	9
Display Component Parameters	10
Pre-initialization Parameters	62
Run-time Parameters	68
Software Image Correction Parameters	72
White Balance Correction Parameters	76
Overlay Text Parameters	78
Dump Rect Parameters	81
IpxDisplay Command Parameters	83
Notifications	89
Translate Flags	91

Fit Modes and Mouse Processing	92
IpxDisplay C++ Class	94
IpxDisplay C-Interface Functions	95
IpxImage Header	11
IpxImageApi Header	13
Image Converter Reference	29
IpxImageConverter C-Interface Functions	102
IpxImageSerializer IpxComponent Header	30
IpxSerializer Parameters	106
IpxImageSerializer C++ Class	110
IpxImageSerializer C-Interface Functions	111
Image Unpacker Reference	31
IpxImageUnpacker C-Interface Functions	116
IpxPixelType Header	32
IpxToolBase Header	46
Error Codes	48
Component Type IDs	49
IpxUserData Header	52

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

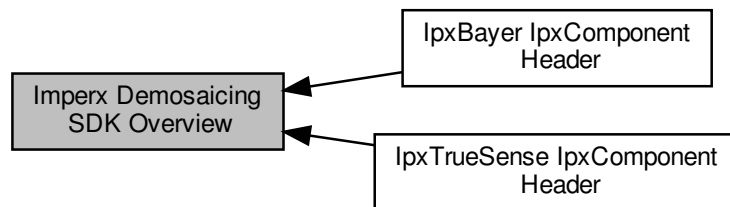
lpxBayer	A Class for lpxBayer modules that contains methods to convert Bayer CFA (Color Filter Array) images	161
lpxComponent	A Class for lpxComponent modules that contains methods for setting/getting/executing Component features	166
lpxDisplay	A Class for lpxDisplay modules that contains methods to display lpxImage images. This class is responsible for displaying video frames and still images	174
lpxImage	Data structure for description of Imperx Image	183
lpxImageConverter	A Class for lpxImageConverter modules that contains methods to convert lpxImage images	186
lpxImageSerializer	lpxComponent to save lpxImage to disk	189
lpxImageUnpacker	lpxComponent to unpack images	195
lpxImgProcessor	Pure virtual base class for image processor	198
lpxPixelTypeDescr	Base type of data for description of lpxImage and other image data types	198
lpxPoint	The lpxPoint structure specifies a point	200
lpxRect	The lpxRect structure defines a rectangle by the coordinates of its upper-left corner and width, height	201
lpxSize	The lpxSize structure specifies a rectangle	202
lpxTrueSense	A Class for lpxTrueSense modules that contains methods to convert lpxImage images	203
lpxUserData	Data structure for description of User Data linked with Imperx Image	208

Chapter 4

Module Documentation

4.1 Imperx Demosaicing SDK Overview

Collaboration diagram for Imperx Demosaicing SDK Overview:



Modules

- [IpxBayer IpxComponent Header](#)
Bayer functions and classes with [IpxComponent](#) features.
- [IpxTrueSense IpxComponent Header](#)
TrueSense functions and classes with [IpxComponent](#) features.

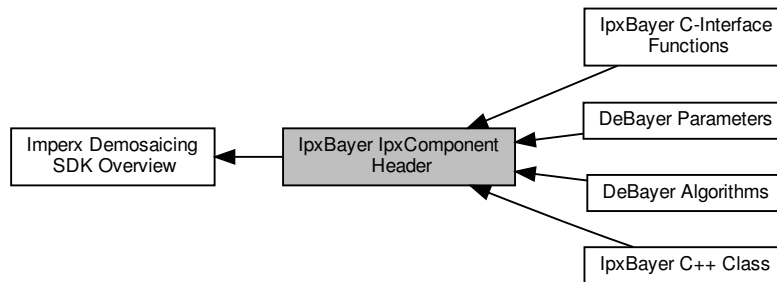
4.1.1 Detailed Description

Imperx Demosaicing SDK dedicated to the conversion of RAW images with Bayer Color Filter Array and Kodak TrueSense Color Filter Array to RGB bitmap. Demosaicing SDK functions allows to convert 8 and 16 bits per pixel RAW images to 3-channel or 4-channel RGB or BGR bitmaps with respectively 8 and 16 bits color depth.

4.2 IpxBayer IpxComponent Header

Bayer functions and classes with [IpxComponent](#) features.

Collaboration diagram for IpxBayer IpxComponent Header:



Modules

- [DeBayer Parameters](#)
Defines for DeBayer Parameters.
- [DeBayer Algorithms](#)
Type of DeBayer Algorithms.
- [IpxBayer C++ Class](#)
C++ Class for [IpxBayer](#).
- [IpxBayer C-Interface Functions](#)
C-interface functions for [IpxBayer](#).

4.2.1 Detailed Description

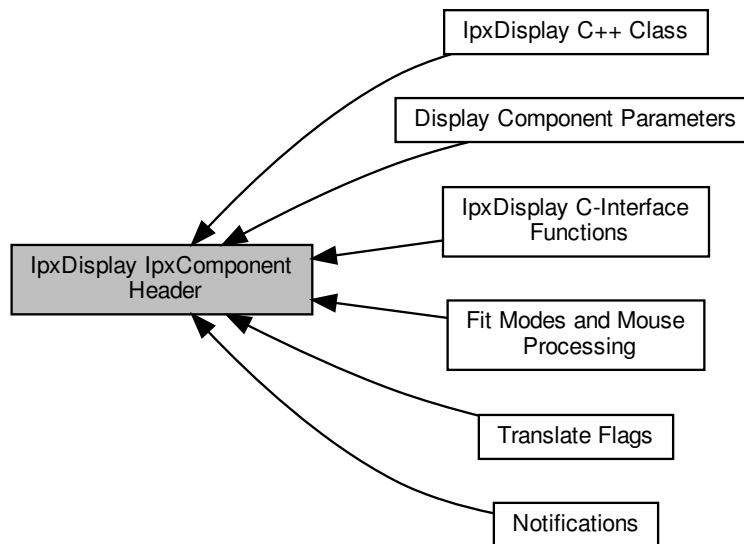
Bayer functions and classes with [IpxComponent](#) features.

This module is responsible for conversion CFA pattern (BAYER) to color image.

4.3 IpxDisplay IpxComponent Header

Display functions and classes with [IpxComponent](#) features.

Collaboration diagram for IpxDisplay IpxComponent Header:



Modules

- [Display Component Parameters](#)
Defines and Macros for Display Component Parameters.
- [Notifications](#)
Defines for Notifications.
- [Translate Flags](#)
Defines for Translate Flags.
- [Fit Modes and Mouse Processing](#)
Defines for Fit Modes and Mouse Processing.
- [IpxDisplay C++ Class](#)
C++ Class for [IpxDisplay](#).
- [IpxDisplay C-Interface Functions](#)
C-interface functions for [IpxDisplay](#).

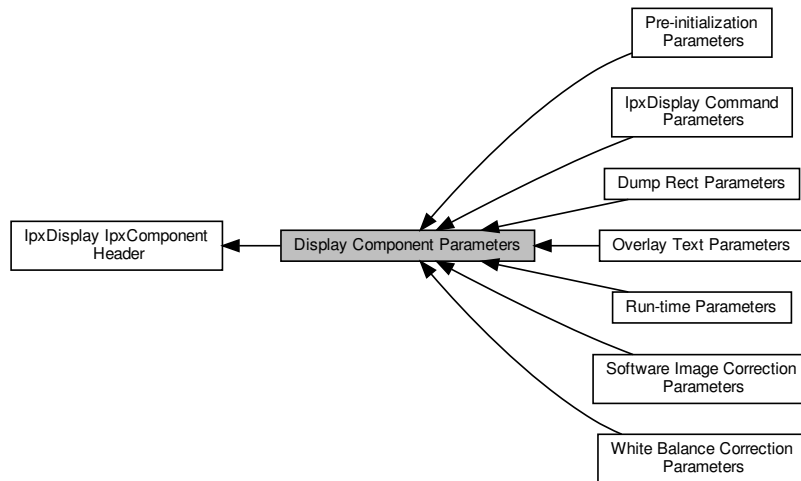
4.3.1 Detailed Description

Display functions and classes with [IpxComponent](#) features.

4.4 Display Component Parameters

Defines and Macros for Display Component Parameters.

Collaboration diagram for Display Component Parameters:



Modules

- [Pre-initialization Parameters](#)
Defines for Pre-Initialization Parameters.
- [Run-time Parameters](#)
Defines for Run-time Parameters (View Management)
- [Software Image Correction Parameters](#)
Defines for Software Image Correction Parameters.
- [White Balance Correction Parameters](#)
Defines for White Balance Correction Parameters.
- [Overlay Text Parameters](#)
Defines for Overlay Text Parameters.
- [Dump Rect Parameters](#)
Defines for Dump Rect Parameters.
- [IpxDisplay Command Parameters](#)
*Defines for *IpxDisplay* Command Parameters.*

4.4.1 Detailed Description

Defines and Macros for Display Component Parameters.

4.5 IpxImage Header

Defines, macros, and functions for `IpxImage`.

Classes

- struct `IpxImage`
Data structure for description of Imperx Image.

Macros

- #define `IPX_IMAGE_MAJOR_VERSION` 2
Defines major version of image data.
- #define `IPX_IMAGE_MINOR_VERSION` 0
Defines minor version of image data.
- #define `IPX_IMAGE_VERSION` ((`IPX_IMAGE_MAJOR_VERSION`<<16)|`IPX_IMAGE_MINOR_VERSION`)
Defines whole version of image data.
- #define `IPX_GET_MAJOR_VERSION`(version) (version>>16)
Gets major version of image data.
- #define `IPX_GET_MINOR_VERSION`(version) ((version<<16)>>16)
Gets minor version of image data.
- #define `IPX_IS_IMAGE_HDR`(iiData) ((iiData) != NULL && ((const IpxData*)(iiData))->nSize == sizeof(`IpxImage`))
Checks whether data is `IpxImage` type.
- #define `IPX_GET_FIRST_PIXEL_DATA`(image) image->imageData
Gets pointer to data of first pixel.
- #define `IPX_GET_PIXEL_DATA`(image, w, h, c)
Gets pointer to data of defined pixel.

Functions

- `IPX_INLINE` bool `IpxInitPixelTypeDescr` (uint32_t pixelType, `IpxPixelTypeDescr` *descr)
Fills descriptor on base value of pixel type.

4.5.1 Detailed Description

Defines, macros, and functions for `IpxImage`.

4.5.2 Function Documentation

4.5.2.1 IpxInitPixelTypeDescr()

```
IPX_INLINE bool IpxInitPixelTypeDescr (
    uint32_t pixelType,
    IpxPixelTypeDescr * descr )
```

Fills descriptor on base value of pixel type.

Parameters

<i>pixelType</i>	Pixel type.
<i>descr</i>	Descriptor of pixel format.

Returns

If the function succeeds, the return value is 'true'. If the function fails, the return value is 'false'.

Here is the call graph for this function:



4.6 lpxImageApi Header

Defines, macros, and functions for lpxImageApi.

Macros

- `#define IPX_FREE(ptr) lpxFree((void**)(ptr))`
That is `lpxFree` wrapper.
- `#define IPX_ALLOC(ptr, size) lpxAlloc((void**)(ptr), size)`
That is `lpxAlloc` wrapper.

Typedefs

- `typedef void *(IPX_CDECL * PAllocFunc) (size_t size)`
Signature of function that allocates a memory block.
- `typedef int(IPX_CDECL * PFreeFunc) (void *ptr)`
Signature of function that deallocates a memory block.

Functions

- `IPXIMAGE_API lpxError lpxSetMemoryManager (PAllocFunc allocFunc, PFreeFunc freeFunc)`
Sets user-defined memory management functions.
- `IPXIMAGE_API lpxError lpxAlloc (void **ptr, size_t size)`
Allocates a memory block.
- `IPXIMAGE_API lpxError lpxFree (void **ptr)`
Deallocates a memory block.
- `IPXIMAGE_API lpxError lpxCreateEmptyImageHeader (lpxImage **image)`
Allocates memory for `lpxImage` header.
- `IPXIMAGE_API lpxError lpxCreateImageHeader (lpxImage **image, lpxSize size, uint32_t pixelType, char *imageData, uint32_t rowSize, int origin)`
Allocates memory for `lpxImage` header and initializes it.
- `IPXIMAGE_API lpxError lpxInitImageHeader (lpxImage *image, lpxSize size, uint32_t pixelType, char *imageData, uint32_t rowSize, int origin)`
Initializes `lpxImage` header.
- `IPXIMAGE_API lpxError lpxCreateImageData (lpxImage *image)`
Allocates memory for `lpxImage` data.
- `IPXIMAGE_API lpxError lpxCreateImage (lpxImage **image, lpxSize size, uint32_t pixelType)`
Allocates memory for `lpxImage` header and data.
- `IPXIMAGE_API lpxError lpxReleaseImageHeader (lpxImage **image)`
Releases memory of `lpxImage` header.
- `IPXIMAGE_API lpxError lpxReleaseImage (lpxImage **image)`
Releases memory of `lpxImage` header and data.
- `IPXIMAGE_API lpxError lpxCloneImage (lpxImage **clone, const lpxImage *image)`

Creates a new copy of `lpxImage`.

- IPXIMAGE_API `lpxError lpxCloneImageExt` (`lpxImage **clone`, const `lpxImage *image`)

Creates a new copy of `lpxImage` with restriction by ROI and COI.

- IPXIMAGE_API `lpxError lpxCopyImageHeader` (`lpxImage *dstImage`, const `lpxImage *srcImage`)

Copies source image header to destination image.

- IPXIMAGE_API `lpxError lpxCopyImage` (`lpxImage *dstImage`, const `lpxImage *srcImage`)

Copy source image to destination image.

- IPXIMAGE_API `lpxError lpxCopyImageChannelChar` (unsigned char *dst, int *dstSize, const `lpxImage *srcImage`, const int channel)

Copies a color channel of source image to a chars array.

- IPXIMAGE_API `lpxError lpxCopyImageChannelShort` (unsigned short *dst, int *dstSize, const `lpxImage *srcImage`, const int channel)

Copies a color channel of source image to a short array.

- IPXIMAGE_API `lpxError lpxCopyImageChannelInt` (int *dst, int *dstSize, const `lpxImage *srcImage`, const int channel)

Copies a color channel of source image to a integer array.

- IPXIMAGE_API `lpxError lpxCopyImageChannelFloat` (float *dst, int *dstSize, const `lpxImage *srcImage`, const int channel)

Copies a color channel of source image to a float array.

4.6.1 Detailed Description

Defines, macros, and functions for `lpxImageApi`.

4.6.2 Typedef Documentation

4.6.2.1 PAllocFunc

```
typedef void*(IPX_CDECL * PAllocFunc) (size_t size)
```

Signature of function that allocates a memory block.

4.6.2.2 PFreeFunc

```
typedef int(IPX_CDECL * PFreeFunc) (void *ptr)
```

Signature of function that deallocates a memory block.

4.6.3 Function Documentation

4.6.3.1 IpxSetMemoryManager()

```
IPXIMAGE_API IpxError IpxSetMemoryManager (
    PAllocFunc allocFunc,
    PFreeFunc freeFunc )
```

Sets user-defined memory management functions.

Parameters

<i>allocFunc</i>	Pointer to the function that allocates a memory block.
<i>freeFunc</i>	Pointer to the function that deallocates a memory block.

Returns

Returns the error code:

- IPX_ERR_OK - Successfully creates a new copy of [IpxImage](#) with restriction by ROI and COI
- IPX_ERR_NULL_POINTER - No srcImage or dstImage.

Note

It sets user-defined memory management functions (substitutors for malloc and free) that will be called by IpxAlloc, IpxFree and higher-level functions (e.g. IpxCreateImage). If the user wants to use the default functions, then it should call IpxSetMemoryManager(NULL, NULL).

For example:

```
IpxSetMemoryManager(NULL, NULL);
. . . . .
void* ptr = NULL;
if (IPX_ERR_OK != IpxAlloc(&ptr, 12345))
{
    IpxError error;
    IpxGetLastError(&error);
    ::_ftprintf_s(file, _T("%s: %d; %s\n"), error.severity, error.code, error.description);
    return error.code;
}
. . . . .
if (IPX_ERR_OK != IpxFree(&ptr))
{
    IpxError error;
    IpxGetLastError(&error);
    ::_ftprintf_s(file, _T("%s: %d; %s\n"), error.severity, error.code, error.description);
    return error.code;
}
```

4.6.3.2 IpxAlloc()

```
IPXIMAGE_API IpxError IpxAlloc (  
    void ** ptr,  
    size_t size )
```

Allocates a memory block.

Parameters

<i>ptr</i>	Pointer to the allocated space.
<i>size</i>	Number of bytes to allocate.

Returns

Returns the error code:

- IPX_ERR_OK - Successfully creates a new copy of [IpxImage](#) with restriction by ROI and COI
- IPX_ERR_NULL_POINTER - No srcImage or dstImage.

Note

This function is <malloc> wrapper. if there is no enough memory, the function raises an error.

See [IpxSetMemoryManager](#) for usage example.

See also

[IpxFree](#)

4.6.3.3 IpxFree()

```
IPXIMAGE_API IpxError IpxFree (  
    void ** ptr )
```

Deallocates a memory block.

Parameters

<i>ptr</i>	Previously allocated memory block to be freed.
------------	--

Returns

Returns the error code:

- IPX_ERR_OK - Successfully creates a new copy of [IpxImage](#) with restriction by ROI and COI
- IPX_ERR_NULL_POINTER - No srcImage or dstImage.

Note

This function is a free wrapper. Passing pointer to NULL pointer is Ok: nothing happens in this case Test requirements

See [IpxSetMemoryManager](#) for usage example.

See also

[IpxAlloc](#)

4.6.3.4 IpxCreateEmptyImageHeader()

```
IPXIMAGE_API IpxError IpxCreateEmptyImageHeader (
    IpxImage ** image )
```

Allocates memory for [IpxImage](#) header.

Parameters

<i>image</i>	Pointer to image header, that will be created.
--------------	--

Returns

Returns the error code:

- IPX_ERR_OK - Successfully creates a new copy of [IpxImage](#) with restriction by ROI and COI
- IPX_ERR_NULL_POINTER - No srcImage or dstImage.

Note

This function don't allocate memory for image data and don't set parameters in image header.

See also

[IpxCreateImageHeader](#)
[IpxReleaseImageHeader](#)
[IpxImage](#)

4.6.3.5 IpxCreateImageHeader()

```
IPXIMAGE_API IpxError IpxCreateImageHeader (
    IpxImage ** image,
    IpxSize size,
    uint32_t pixelType,
    char * imageData,
    uint32_t rowSize,
    int origin )
```

Allocates memory for [IpxImage](#) header and initializes it.

Parameters

<i>image</i>	Pointer to image header, that will be created.
<i>size</i>	Horizontal and vertical size of image.
<i>pixelType</i>	Type of image pixel.
<i>imageData</i>	Pointer to image data.
<i>rowSize</i>	Size of image row in bytes.
<i>origin</i>	Origin of image coordinate system.

Returns

If the function succeeds, the return value is 0. If the function fails, the return value is a non-zero code.

Note

This function uses a given image data. For example:

```
IpxImage* img = NULL;
IpxSize size;
size.height = 480;
size.width = 640;
if (IPX_ERR_OK == IpxCreateImageHeader(&img, size,
    II_PIX_RGB8, NULL, 0))
{
    IpxError error;
    IpxGetLastError(&error);
    return error.code;
}
. . . . .
IpxReleaseImage(&img);

//
```

See also

[IpxCreateImage](#)
[IpxReleaseImage](#)
[IpxImage](#)

4.6.3.6 IpxInitImageHeader()

```
IPXIMAGE_API IpxError IpxInitImageHeader (
    IpxImage * image,
    IpxSize size,
    uint32_t pixelType,
    char * imageData,
    uint32_t rowSize,
    int origin )
```

Initializes [IpxImage](#) header.

Parameters

<i>image</i>	Pointer to image header.
<i>size</i>	Horizontal and vertical size of image.
<i>pixelType</i>	Type of image pixel.
<i>imageData</i>	Pointer to image data.
<i>rowSize</i>	Size of image row in bytes.
<i>origin</i>	Origin of image coordinate system.

Returns

Returns the error code:

- IPX_ERR_OK - Successfully creates a new copy of [IpxImage](#) with restriction by ROI and COI
- IPX_ERR_NULL_POINTER - No srcImage or dstImage.

Note

This function uses a given image data. For example:

```
IpxImage* img = NULL;
if (IPX_ERR_OK == IpxCreateEmptyImageHeader (&img))
{
    IpxError error;
    IpxGetLastError(&error);
    return error.code;
}
IpxSize size;
size.height = 480;
size.width = 640;
if (IPX_ERR_OK == IpxInitImageHeader(img, size,
    II_PIX_RGB8, NULL, 0))
{
    IpxReleaseImageHeader (&img);
    IpxError error;
    IpxGetLastError(&error);
    return error.code;
}
. . . . .
IpxReleaseImage (&img);
```

See also

[IpxCreateImage](#)
[IpxReleaseImage](#)
[IpxImage](#)

4.6.3.7 IpxCreateImageData()

```
IPXIMAGE_API IpxError IpxCreateImageData (
    IpxImage * image )
```

Allocates memory for [IpxImage](#) data.

Parameters

<i>image</i>	Pointer to image header.
--------------	--------------------------

Returns

Returns the error code:

- IPX_ERR_OK - Successfully creates a new copy of [IpxImage](#) with restriction by ROI and COI
- IPX_ERR_NULL_POINTER - No srcImage or dstImage.

Note

This function allocates memory for image data in accordance to parameters of image header.

See also

[IpxCreateImageHeader](#)
[IpxReleaseImage](#)
[IpxImage](#)

4.6.3.8 IpxCreateImage()

```
IPXIMAGE_API IpxError IpxCreateImage (
    IpxImage ** image,
    IpxSize size,
    uint32_t pixelType )
```

Allocates memory for [IpxImage](#) header and data.

Parameters

<i>image</i>	Pointer to IpxImage , that will be created.
<i>size</i>	Horizontal and vertical size of image.
<i>pixelType</i>	Type of image pixel.

Returns

Returns the error code:

- `IPX_ERR_OK` - Successfully creates a new copy of [IpxImage](#) with restriction by ROI and COI
- `IPX_ERR_NULL_POINTER` - No srcImage or dstImage.

Note

This function allocates memory for image header and data.

See also

[IpxCreateImageHeader](#)
[IpxReleaseImage](#)
[IpxImage](#)

4.6.3.9 IpxReleaseImageHeader()

```
IPXIMAGE_API IpxError IpxReleaseImageHeader (
    IpxImage ** image )
```

Releases memory of [IpxImage](#) header.

Parameters

<i>image</i>	Pointer to IpxImage image, that will be created.
--------------	--

Returns

Returns the error code:

- `IPX_ERR_OK` - Successfully creates a new copy of [IpxImage](#) with restriction by ROI and COI
- `IPX_ERR_NULL_POINTER` - No srcImage or dstImage.

Note

This function deallocates memory of image header, but not memory of image data.

See also

[IpxCreateImageHeader](#)
[IpxReleaseImage](#)
[IpxImage](#)

4.6.3.10 IpxReleaseImage()

```
IPXIMAGE_API IpxError IpxReleaseImage (
    IpxImage ** image )
```

Releases memory of [IpxImage](#) header and data.

Parameters

<i>image</i>	Pointer to IpxImage image, that will be created.
--------------	--

Returns

Returns the error code:

- IPX_ERR_OK - Successfully creates a new copy of [IpxImage](#) with restriction by ROI and COI
- IPX_ERR_NULL_POINTER - No srcImage or dstImage.

Note

This function deallocates memory of image header and data.

See also

[IpxCreateImageHeader](#)
[IpxCreateImage](#)
[IpxImage](#)

4.6.3.11 IpxCloneImage()

```
IPXIMAGE_API IpxError IpxCloneImage (
    IpxImage ** clone,
    const IpxImage * image )
```

Creates a new copy of [IpxImage](#).

Parameters

<i>clone</i>	Pointer to new image.
<i>image</i>	Pointer to source image.

Returns

Returns the error code:

- `IPX_ERR_OK` - Successfully creates a new copy of [IpxImage](#) with restriction by ROI and COI
- `IPX_ERR_NULL_POINTER` - No srcImage or dstImage.

Note

This function allocates memory for new image and copies source image to it.

See also

[IpxCloneImageExt](#)
[IpxCopyImage](#)
[IpxCreateImage](#)
[IpxImage](#)

4.6.3.12 IpxCloneImageExt()

```
IPXIMAGE_API IpxError IpxCloneImageExt (
    IpxImage ** clone,
    const IpxImage * image )
```

Creates a new copy of [IpxImage](#) with restriction by ROI and COI.

Parameters

<i>clone</i>	Pointer to new image.
<i>image</i>	Pointer to source image.

Returns

Returns the error code:

- `IPX_ERR_OK` - Successfully creates a new copy of [IpxImage](#) with restriction by ROI and COI
- `IPX_ERR_NULL_POINTER` - No srcImage or dstImage.

Note

This function allocates memory for the new image and copies the image data that is restricted by ROI and COI.

See also

[IpxCloneImage](#)
[IpxCopyImage](#)
[IpxCreateImage](#)
[IpxImage](#)

4.6.3.13 IpxCopyImageHeader()

```
IPXIMAGE_API IpxError IpxCopyImageHeader (
    IpxImage * dstImage,
    const IpxImage * srcImage )
```

Copies source image header to destination image.

Parameters

<i>dstImage</i>	Pointer to destination image.
<i>srcImage</i>	Pointer to source image.

Returns

Returns the error code:

- IPX_ERR_OK - Successfully copies source image header to destination image
- IPX_ERR_NULL_POINTER - No srcImage or dstImage.

Note

This function copies source image header to destination image.

See also

IpxCloneMatrix
IpxCreateMatrixFromImage

4.6.3.14 IpxCopyImage()

```
IPXIMAGE_API IpxError IpxCopyImage (
    IpxImage * dstImage,
    const IpxImage * srcImage )
```

Copy source image to destination image.

Parameters

<i>dstImage</i>	Pointer to destination image.
<i>srcImage</i>	Pointer to source image.

Returns

Returns the error code:

- `IPX_ERR_OK` - Successfully copies source image to destination image
- `IPX_ERR_NULL_POINTER` - No `srcImage` or `dstImage`.
- `IPX_ERR_INVALID_ARGUMENT` - invalid argument. For example, `dstSize` is less than `srcImage` size

Note

This function checks coincidence of size and pixel type in source and destination.

See also

[IpxCopyImage](#)
[IpxCreateImage](#)
[IpxImage](#)

4.6.3.15 IpxCopyImageChannelChar()

```
IPXIMAGE_API IpxError IpxCopyImageChannelChar (
    unsigned char * dst,
    int * dstSize,
    const IpxImage * srcImage,
    const int channel )
```

Copies a color channel of source image to a chars array.

Parameters

<i>dst</i>	Pointer to destination array.
<i>dstSize</i>	Pointer to the value of array size.
<i>srcImage</i>	Pointer to source image.
<i>channel</i>	Channel number of source image.

Returns

Returns the error code:

- `IPX_ERR_OK` - Successfully copies a color channel of source image to a char array
- `IPX_ERR_NULL_POINTER` - Unable to copy the image Channels.
- `IPX_ERR_INVALID_ARGUMENT` - invalid argument. For example, `dstSize` is less than `srcImage` size

Note

This function copies a color channel of source image to a char array. If `dst = NULL` or `*dstSize = 0`, then the function returns the required size of destination array in `dstSize` value. Otherwise, the function returns the size of image row in pixels.

See also

[IpxCopyImage](#)
[IpxCreateImage](#)
[IpxImage](#)

4.6.3.16 IpxCopyImageChannelShort()

```
IPXIMAGE_API IpxError IpxCopyImageChannelShort (
    unsigned short * dst,
    int * dstSize,
    const IpxImage * srcImage,
    const int channel )
```

Copies a color channel of source image to a short array.

Parameters

<i>dst</i>	Pointer to destination array.
<i>dstSize</i>	Pointer to the value of array size.
<i>srcImage</i>	Pointer to source image.
<i>channel</i>	Channel number of source image.

Returns

Returns the error code:

- `IPX_ERR_OK` - Successfully copies a color channel of source image to a short array
- `IPX_ERR_NULL_POINTER` - Unable to copy the image Channels.
- `IPX_ERR_INVALID_ARGUMENT` - invalid argument. For example, `dstSize` is less than `srcImage` size

Note

This function copies a color channel of source image to a shorts array. If `dst = NULL` or `*dstSize = 0`, then the function returns the required size of the destination array in `dstSize` value. Otherwise, the function returns the size of image row in pixels.

See also

[IpxCopyImage](#)
[IpxCreateImage](#)
[IpxImage](#)

4.6.3.17 IpxCopyImageChannelInt()

```
IPXIMAGE_API IpxError IpxCopyImageChannelInt (
    int * dst,
    int * dstSize,
    const IpxImage * srcImage,
    const int channel )
```

Copies a color channel of source image to a integer array.

Parameters

<i>dst</i>	Pointer to destination array.
<i>dstSize</i>	Pointer to the value of array size.
<i>srcImage</i>	Pointer to source image.
<i>channel</i>	Channel number of source image.

Returns

Returns the error code:

- `IPX_ERR_OK` - Successfully copies a color channel of source image to an integer array
- `IPX_ERR_NULL_POINTER` - Unable to copy the image Channels.
- `IPX_ERR_INVALID_ARGUMENT` - invalid argument. For example, `dstSize` is less than `srcImage` size

Note

This function copies a color channel of source image to an integer array. If `dst = NULL` or `*dstSize = 0`, then the function returns the required size of the destination array in `dstSize` value. Otherwise, the function returns the size of the image row in pixels.

See also

[IpxCopyImage](#)
[IpxCreateImage](#)
[IpxImage](#)

4.6.3.18 IpxCopyImageChannelFloat()

```
IPXIMAGE_API IpxError IpxCopyImageChannelFloat (
    float * dst,
    int * dstSize,
    const IpxImage * srcImage,
    const int channel )
```

Copies a color channel of source image to a float array.

Parameters

<i>dst</i>	Pointer to destination array.
<i>dstSize</i>	Pointer to the value of array size.
<i>srcImage</i>	Pointer to source image.
<i>channel</i>	Channel number of source image.

Returns

Returns the error code:

- `IPX_ERR_OK` - Successfully copies a color channel of source image to a float array
- `IPX_ERR_NULL_POINTER` - Unable to copy the image Channels.
- `IPX_ERR_INVALID_ARGUMENT` - invalid argument. For example, `dstSize` is less than `srcImage` size

Note

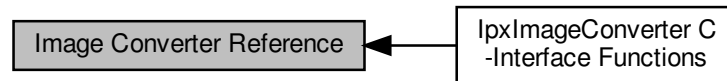
This function copies a color channel of source image to a floats array. If `dst = NULL` or `*dstSize = 0`, then the function returns the required size of destination array in `dstSize` value. Otherwise, the function returns the size of the image row in pixels.

See also

[lpxCopyImage](#)
[lpxCreateImage](#)
[lpxImage](#)

4.7 Image Converter Reference

Collaboration diagram for Image Converter Reference:



Modules

- [lpxImageConverter C-Interface Functions](#)

Classes

- class [lpxImageConverter](#)
A Class for [lpxImageConverter](#) modules that contains methods to convert [lpxImage](#) images.

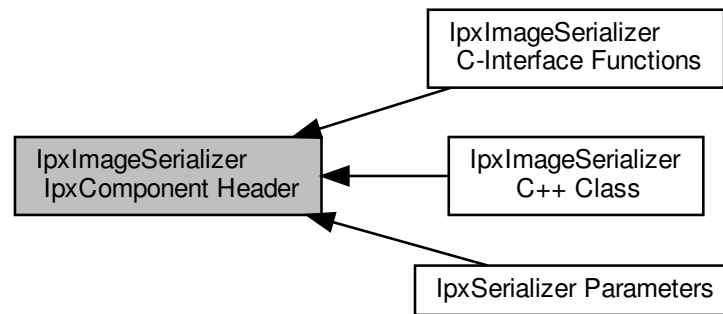
4.7.1 Detailed Description

The following items are exists in Image Converter

4.8 IpxImageSerializer IpxComponent Header

[IpxImageSerializer](#) vunctions and classes with [IpxComponent](#) features.

Collaboration diagram for IpxImageSerializer IpxComponent Header:



Modules

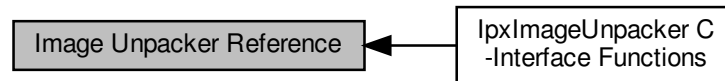
- [IpxSerializer Parameters](#)
Defines for `IpxSerializer Parameters`.
- [IpxImageSerializer C++ Class](#)
C++ Class for `IpxImageSerializer`.
- [IpxImageSerializer C-Interface Functions](#)
C-interface functions for `IpxImageSerializer`.

4.8.1 Detailed Description

[IpxImageSerializer](#) vunctions and classes with [IpxComponent](#) features.

4.9 Image Unpacker Reference

Collaboration diagram for Image Unpacker Reference:



Modules

- [lpxImageUnpacker C-Interface Functions](#)

Classes

- class [lpxImageUnpacker](#)
[lpxComponent](#) to unpack images.

4.9.1 Detailed Description

The following items exist in [lpxImageUnpacker](#)

4.10 lpxPixelType Header

Defines, macros for lpxPixelTypes.

Macros

- `#define II_PIXEL_ALIGNMENT_MASK 0x0000FF00`
Mask to get pixel alignment.
- `#define II_PIXEL_ALIGNMENT_PACK_MASK 0x0000F000`
Mask to get pixel packing.
- `#define II_PIXEL_COLOR_MASK 0xFF000000`
Mask to get pixel chromaticity.
- `#define II_PIXEL_SIZE_MASK 0x00FF0000`
Mask to get pixel chromaticity.
- `#define II_PIXEL_SIZE_SHIFT 16`
Mask to get shift of pixel size.
- `#define II_PIXEL_ID_MASK 0x000000FF`
Mask to get pixel ID value.
- `#define II_GET_ROW_SIZE(pixType, width) lpxGetRowSize(pixType, width)`
Returns aligned row size for defined pixel type and number of pixels in row.
- `#define II_GET_PIXEL_ALIGNMENT(pixType) (pixType & II_PIXEL_ALIGNMENT_MASK)`
Returns pixel alignment for defined pixel type.
- `#define II_GET_PIXEL_CHROMATICITY(pixType) (pixType & II_PIXEL_COLOR_MASK)`
Returns pixel chromaticity for defined pixel type.
- `#define II_IS_COLOR_PIXEL(pixType) ((pixType & II_PIXEL_COLOR_MASK) == II_PIX_COLOR)`
Returns 'true' if pixel is colored.
- `#define II_IS_CUSTOM_PIXEL(pixType) ((pixType & II_PIXEL_COLOR_MASK) == II_PIX_CUSTOM)`
Returns 'true' if pixel is custom.
- `#define II_GET_PIXEL_BITS_SIZE(pixType) ((pixType & II_PIXEL_SIZE_MASK) >> II_PIXEL_SIZE_SHIFT)`
Returns size of pixel in bits for defined pixel type.
- `#define II_GET_PIXEL_ID(pixType) (pixType & II_PIXEL_ID_MASK)`
Returns identifier of pixel type for defined pixel type.
- `#define II_GET_PIXEL_TYPE_INDEX(pixType) ((pixType & II_PIXEL_ID_MASK) - 1)`
Returns index of pixel type for defined pixel type.
- `#define II_GET_IMAGE_SIZE(pixType, width, height) (height * II_GET_ROW_SIZE(pixType, width))`
Returns aligned image size for defined pixel type, width and height.
- `#define II_IS_PACKED_PIXEL(pixType) (II_PIXEL_ALIGNMENT_PACK_MASK & pixType)`
Returns 'true' if pixel is packed.
- `#define II_IS_PACKED_PIXEL_PFNC(pixType) ((II_PIXEL_ALIGNMENT_PACK_MASK & pixType) == II_ALIGN_IGNORED_PACKED_PFNC)`
Returns 'true' if pixel is packed, accorsing PFNC scheme.
- `#define II_IS_PACKED_PIXEL_GEV(pixType) ((II_PIXEL_ALIGNMENT_PACK_MASK & pixType) == II_ALIGN_IGNORED_PACKED_GEV)`
Returns 'true' if pixel is packed, accorsing GEV scheme.
- `#define II_IS_BAYER_CFA_PIXEL(__pixType__) (II_GET_PIXEL_CHROMATICITY(__pixType__) == II_PIXEL_CHROMATICITY_BAYER_CFA)`

Returns 'true' if pixel type is Bayer CFA pattern.

- #define `II_IS_SPARSE_CFA_PIXEL(__pixType__)` (`II_GET_PIXEL_CHROMATICITY(__pixType__)` == `II_PIX_SPARSE_CFA`)

Returns 'true' if pixel type is Sparse CFA pattern.

- #define `II_IS_MONO_PIXEL(__pixType__)` (`II_GET_PIXEL_CHROMATICITY(__pixType__)` == `II_PIX_MONO`)

Returns 'true' if pixel type is Monochrome.

- #define `II_IS_COLOR_RGB_PIXEL(__pixType__)` (`II_GET_PIXEL_CHROMATICITY(__pixType__)` == `II_PIX_COLOR`)

Returns 'true' if pixel type is Color RGB or BGR.

Enumerations

- enum `II_PIXEL_ALIGNMENT` : uint32_t {
`II_ALIGN_8` = 0x00000100, `II_ALIGN_10` = 0x00000200, `II_ALIGN_12` = 0x00000300, `II_ALIGN_14` = 0x00000400,
`II_ALIGN_16` = 0x00000500, `II_ALIGN_10_PACKED_GEV` = 0x00001200, `II_ALIGN_12_PACKED_GEV` = 0x00001300, `II_ALIGN_10_PACKED_PFNC` = 0x00002200,
`II_ALIGN_12_PACKED_PFNC` = 0x00002300, `II_ALIGN_8_PACKED_FLEX` = 0x00003100, `II_ALIGN_10_PACKED_FLEX` = 0x00003200, `II_ALIGN_12_PACKED_FLEX` = 0x00003300 }

Define pixel alignment.

- enum `II_PIXEL_CHROMATICITY` : uint32_t {
`II_PIX_MONO` = 0x01000000, `II_PIX_COLOR` = 0x02000000, `II_PIX_BAYER_CFA` = 0x03000000, `II_PIX_SPARSE_CFA` = 0x04000000,
`II_PIX_YUV` = 0x05000000, `II_PIX_CUSTOM` = 0x80000000 }

Define pixel chromaticity.

- enum `II_PIXEL_BITS` : uint32_t {
`II_PIX_OCCUPY_1_BIT` = 0x00010000, `II_PIX_OCCUPY_2_BIT` = 0x00020000, `II_PIX_OCCUPY_4_BIT` = 0x00040000, `II_PIX_OCCUPY_8_BIT` = 0x00080000,
`II_PIX_OCCUPY_10_BIT` = 0x000A0000, `II_PIX_OCCUPY_12_BIT` = 0x000C0000, `II_PIX_OCCUPY_16_BIT` = 0x00100000, `II_PIX_OCCUPY_20_BIT` = 0x00140000,
`II_PIX_OCCUPY_24_BIT` = 0x00180000, `II_PIX_OCCUPY_32_BIT` = 0x00200000, `II_PIX_OCCUPY_36_BIT` = 0x00240000, `II_PIX_OCCUPY_48_BIT` = 0x00300000 }

Define effective number of bits occupied by the pixel (including padding).

- enum `II_PIXEL_TYPE_DEFINES` : uint32_t {
`II_PIX_MONO8` = (`II_PIX_MONO` | `II_PIX_OCCUPY_8_BIT` | `II_ALIGN_8` | (`II_MONO_ID_MIN`+0x00)) , `II_PIX_BAYGR8` = (`II_PIX_BAYER_CFA` | `II_PIX_OCCUPY_8_BIT` | `II_ALIGN_8` | (`II_BAYER_CFA_ID_MIN`+0x00)) ,
`II_PIX_TS_BGGR_WBBW0_8` = (`II_PIX_SPARSE_CFA` | `II_PIX_OCCUPY_8_BIT` | `II_ALIGN_8` | (`II_SPARSE_CFA_ID_MIN`+0x00)) , `II_PIX_RGB8` = (`II_PIX_COLOR` | `II_PIX_OCCUPY_24_BIT` | `II_ALIGN_8` | (`II_RGB_ID_MIN`+0x00)) ,
`II_PIX_YUV422_8_UYVY` = (`II_PIX_YUV` | `II_PIX_OCCUPY_16_BIT` | `II_ALIGN_8` | (`II_YUV_ID_MIN`+0x00)) ,
`II_PIX_NONE_TYPE` = 0 }

Definition of Pixel Types for Images which are processed in `lpxImage`.

Functions

- `IPX_INLINE uint32_t lpxGetRowSize` (uint32_t pixType, uint32_t width)
Returns row size for defined pixel type and number of pixels in row.
- `IPX_INLINE uint32_t lpxGetRowSizeUnaligned` (uint32_t pixType, uint32_t width)
Returns the size of unaligned row for defined pixel type and number of pixels.

- IPX_INLINE int32_t [lpxGetPixelTypesNumber](#) ()
Returns the number of Pixel Types (Color Models) that are supported by this header file.
- IPX_INLINE bool [lpxIsPixelFormat](#) (uint32_t pixelType)
Defines whether the number is the pixel type.
- IPX_INLINE bool [lpxIsGroup](#) (char *groupName, uint32_t pixelType)
Defines whether the pixel type is a member of a group.
- IPX_INLINE const lpxColorModelDescription * [lpxGetColorModelDescription](#) (uint32_t pixelType)
Defines color model descriptor by Pixel Type.
- IPX_INLINE const lpxColorModelDescription * [lpxGetColorModelDescr](#) (uint32_t index)
Defines color model descriptor by an index.
- IPX_INLINE uint32_t [lpxGetPixelFormat](#) (char *colorModelName)
Defines pixel type by name of color model.
- IPX_INLINE const char * [lpxGetColorModelName](#) (uint32_t pixelType)
Defines name of color model by pixel type.
- IPX_INLINE const char * [lpxGetChannelSequence](#) (uint32_t pixelType)
Defines sequence of channels.
- IPX_INLINE int32_t [lpxGetChannelsNumber](#) (uint32_t pixelType)
Defines number of channels.
- IPX_INLINE int32_t [lpxGetChannelsDepth](#) (uint32_t pixelType)
Defines depth of color channel.
- IPX_INLINE int32_t [lpxGetStartPosition](#) (uint32_t pixelType)
Defines start position in a CFA.
- IPX_INLINE int32_t [lpxGetChannelIndex](#) (uint32_t pixelType, int16_t chName)
Defines index of color channel.
- IPX_INLINE [lpxError](#) [lpxCheckChannelNames](#) (uint32_t pixelType, int16_t *chNames, int32_t channels)
Checks channel names.
- IPX_INLINE [lpxError](#) [lpxConvertChannelStr](#) (char *nameStr, const char *sep, int16_t *chNames, int32_t channels)
Converts string to array of channel names.
- IPX_INLINE int16_t [lpxGetChannelName](#) (uint32_t pixelType, int32_t chnIdx)
Gets channel name.

4.10.1 Detailed Description

Defines, macros for lpxPixelFormat.

lpxPixelFormat Headers

4.10.2 Enumeration Type Documentation

4.10.2.1 II_PIXEL_ALIGNMENT

```
enum II_PIXEL_ALIGNMENT : uint32_t
```

Define pixel alignment.

Note

Pixel alignment defines order of bits placement.

See also

[II_PIXEL_TYPE_DEFINES](#)

Enumerator

II_ALIGN_8	8-bit unsigned. Value range: 0 to 255
II_ALIGN_10	10-bit unsigned. Value range: 0 to 1023
II_ALIGN_12	12-bit unsigned. Value range: 0 to 4095
II_ALIGN_14	14-bit unsigned. Value range: 0 to 16383
II_ALIGN_16	16-bit unsigned. Value range: 0 to 65535
II_ALIGN_10_PACKED_GEV	10-bit unsigned. Value range: 0 to 1023 - GigE Vision Mono10Packed, BayerXX10Packed alignment
II_ALIGN_12_PACKED_GEV	12-bit unsigned. Value range: 0 to 4095 - GigE Vision Mono12Packed, BayerXX12Packed alignment
II_ALIGN_10_PACKED_PFNC	10-bit unsigned. Value range: 0 to 1023 - PFNC Mono10p, BayerXX10p alignment, used in U3V
II_ALIGN_12_PACKED_PFNC	12-bit unsigned. Value range: 0 to 4095 - PFNC Mono12p, BayerXX12p alignment, used in U3V
II_ALIGN_8_PACKED_FLEX	8-bit unsigned. Value range: 0 to 255 - Alignment scheme, used in Framelink Express grabber
II_ALIGN_10_PACKED_FLEX	10-bit unsigned. Value range: 0 to 1023 - Alignment scheme, used in Framelink Express grabber
II_ALIGN_12_PACKED_FLEX	12-bit unsigned. Value range: 0 to 4095 - Alignment scheme, used in Framelink Express grabber

4.10.2.2 II_PIXEL_CHROMATICITY

```
enum II_PIXEL_CHROMATICITY : uint32_t
```

Define pixel chromaticity.

Note

Pixel chromaticity defines number of color channels in an image.

See also

[II_PIXEL_TYPE_DEFINES](#)

Enumerator

II_PIX_MONO	Monochrome pixel.
II_PIX_COLOR	Colored RGB pixel.
II_PIX_BAYER_CFA	Bayer CFA pixel.
II_PIX_SPARSE_CFA	Sparse TRUESENSE CFA pixel.
II_PIX_YUV	YUV, YCbCr pixel.
II_PIX_CUSTOM	Custom defined pixel type.

4.10.2.3 II_PIXEL_BITS

```
enum II_PIXEL_BITS : uint32_t
```

Define effective number of bits occupied by the pixel (including padding).

Note

This value can be used to quickly compute the amount of memory required to store an image using pixel type.

See also

[II_PIXEL_TYPE_DEFINES](#)

Enumerator

II_PIX_OCCUPY_1_BIT	Pixel size: 1 bits
II_PIX_OCCUPY_2_BIT	Pixel size: 2 bits
II_PIX_OCCUPY_4_BIT	Pixel size: 4 bits
II_PIX_OCCUPY_8_BIT	Pixel size: 8 bits
II_PIX_OCCUPY_10_BIT	Pixel size: 10 bits
II_PIX_OCCUPY_12_BIT	Pixel size: 12 bits
II_PIX_OCCUPY_16_BIT	Pixel size: 16 bits
II_PIX_OCCUPY_20_BIT	Pixel size: 20 bits

Enumerator

II_PIX_OCCUPY_24_BIT	Pixel size: 24 bits
II_PIX_OCCUPY_32_BIT	Pixel size: 32 bits
II_PIX_OCCUPY_36_BIT	Pixel size: 36 bits
II_PIX_OCCUPY_48_BIT	Pixel size: 48 bits

4.10.2.4 II_PIXEL_TYPE_DEFINES

```
enum II_PIXEL_TYPE_DEFINES : uint32_t
```

Definition of Pixel Types for Images which are processed in [lpxImage](#).

Note

Each pixel type is represented by a 32-bit value. The upper 8-bit indicates the pixel chromaticity. The second upper 8-bit indicates the number of bit occupied by a pixel (including any padding). This can be used to quickly compute the amount of memory required to store an image using pixel type. Next 8-bit indicates pixel alignment that defines order of bits placement. Lower 8-bit indicates the pixel type identifier (pixel ID). Thus, pixel type contains main information about pixel structure. But pixel type don't define such parameters as color depth and channel order.

See also

[II_PIXEL_CHROMATICITY](#)
[II_PIXEL_BITS](#)
[II_PIXEL_ALIGNMENT](#)

Enumerator

II_PIX_MONO8	That and next types define grayscale pixels
II_PIX_BAYGR8	That and next types define Bayer pixels
II_PIX_TS_BGGR_WBBW0↔ _8	That and next types define Sparse CFA pixels
II_PIX_RGB8	That and next types define RGB-BGR pixels
II_PIX_YUV422_8_UYVY	That and next types define YUV and TCbCr packed pixels
II_PIX_NONE_TYPE	The label for undefined pixel type

4.10.3 Function Documentation

4.10.3.1 IpxGetRowSize()

```
IPX_INLINE uint32_t IpxGetRowSize (
    uint32_t pixType,
    uint32_t width )
```

Returns row size for defined pixel type and number of pixels in row.

Parameters

<i>pixType</i>	Pixel type.
<i>width</i>	Number of pixels in row.

Returns

The return value is row size.

Note

Row size is aligned for effective memory using.

4.10.3.2 IpxGetRowSizeUnaligned()

```
IPX_INLINE uint32_t IpxGetRowSizeUnaligned (
    uint32_t pixType,
    uint32_t width )
```

Returns the size of unaligned row for defined pixel type and number of pixels.

Parameters

<i>pixType</i>	Pixel type.
<i>width</i>	Number of pixels in row.

Returns

The return value is row size.

Note

Row size is aligned for effective memory using.

4.10.3.3 IpxGetPixelTypesNumber()

```
IPX_INLINE int32_t IpxGetPixelTypesNumber ( )
```

Returns the number of Pixel Types (Color Models) that are supported by this header file.

Returns

The return value is the number of Pixel Types.

4.10.3.4 IpxIsPixelFormat()

```
IPX_INLINE bool IpxIsPixelFormat (
    uint32_t pixelType )
```

Defines whether the number is the pixel type.

Parameters

<i>pixelType</i>	Pixel type.
------------------	-------------

Returns

The return value is 'true' if pixelType is pixel type.

4.10.3.5 IpxIsGroup()

```
IPX_INLINE bool IpxIsGroup (
    char * groupName,
    uint32_t pixelType )
```

Defines whether the pixel type is a member of a group.

Parameters

<i>groupName</i>	Group name or some substring in Color Model Name.
<i>pixelType</i>	Pixel type.

Returns

The return value is 'true' if pixType is a member of group.

4.10.3.6 IpxGetColorModelDescription()

```
IPX_INLINE const IpxColorModelDescription* IpxGetColorModelDescription (
    uint32_t pixelType )
```

Defines color model descriptor by Pixel Type.

Parameters

<i>pixelType</i>	Pixel type.
------------------	-------------

Returns

The return value is pointer to color model descriptor.

Here is the caller graph for this function:

**4.10.3.7 IpxGetColorModelDescr()**

```
IPX_INLINE const IpxColorModelDescription* IpxGetColorModelDescr (
    uint32_t index )
```

Defines color model descriptor by an index.

Parameters

<i>index</i>	descriptor index.
--------------	-------------------

Returns

The return value is pointer to color model descriptor.

4.10.3.8 IpxGetPixelFormat()

```
IPX_INLINE uint32_t IpxGetPixelFormat (  
    char * colorModelName )
```

Defines pixel type by name of color model.

Parameters

<i>colorModelName</i>	Name of color model.
-----------------------	----------------------

Returns

The return value is pixel type.

4.10.3.9 IpxGetColorModelName()

```
IPX_INLINE const char* IpxGetColorModelName (  
    uint32_t pixelType )
```

Defines name of color model by pixel type.

Parameters

<i>pixelType</i>	Pixel type.
------------------	-------------

Returns

The return value is name of color model.

4.10.3.10 IpxGetChannelSequence()

```
IPX_INLINE const char* IpxGetChannelSequence (
    uint32_t pixelType )
```

Defines sequence of channels.

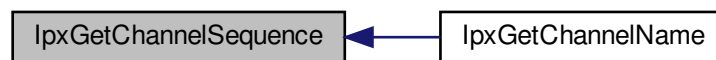
Parameters

<i>pixelType</i>	Pixel type.
------------------	-------------

Returns

The return value is sequence of channels.

Here is the caller graph for this function:



4.10.3.11 IpxGetChannelsNumber()

```
IPX_INLINE int32_t IpxGetChannelsNumber (
    uint32_t pixelType )
```

Defines number of channels.

Parameters

<i>pixelType</i>	Pixel type.
------------------	-------------

Returns

The return value is number of channels.

4.10.3.12 IpxGetChannelsDepth()

```
IPX_INLINE int32_t IpxGetChannelsDepth (
    uint32_t pixelType )
```

Defines depth of color channel.

Parameters

<i>pixelType</i>	Pixel type.
------------------	-------------

Returns

The return value is depth of color channel.

4.10.3.13 IpxGetStartPosition()

```
IPX_INLINE int32_t IpxGetStartPosition (
    uint32_t pixelType )
```

Defines start position in a CFA.

Parameters

<i>pixelType</i>	Pixel type.
------------------	-------------

Returns

The return value is start position in a CFA.

4.10.3.14 IpxGetChannelIndex()

```
IPX_INLINE int32_t IpxGetChannelIndex (
    uint32_t pixelType,
    int16_t chName )
```

Defines index of color channel.

Parameters

<i>pixelType</i>	Pixel type.
<i>chName</i>	Channel name.

Returns

The return value is index of color channel.

4.10.3.15 IpxCheckChannelNames()

```
IPX_INLINE IpxError IpxCheckChannelNames (
    uint32_t pixelType,
    int16_t * chNames,
    int32_t channels )
```

Checks channel names.

Parameters

<i>pixelType</i>	Pixel type.
<i>chNames</i>	Array of channel names.
<i>channels</i>	Number of checked names.

Returns

If the function succeeds, the return value is 0. If the function fails, the return value is -1.

4.10.3.16 IpxConvertChannelStr()

```
IPX_INLINE IpxError IpxConvertChannelStr (
    char * nameStr,
    const char * sep,
    int16_t * chNames,
    int32_t * channels )
```

Converts string to array of channel names.

Parameters

<i>nameStr</i>	String of channel names.
<i>sep</i>	Separator of channel names in the string.
<i>chNames</i>	Array of channel names.
<i>channels</i>	Number of channel names.

Returns

If the function succeeds, the return value is 0. If the function fails, the return value is -1.

Here is the caller graph for this function:

**4.10.3.17 IpxGetChannelName()**

```
IPX_INLINE int16_t IpxGetChannelName (  
    uint32_t pixelType,  
    int32_t chnlIndx )
```

Gets channel name.

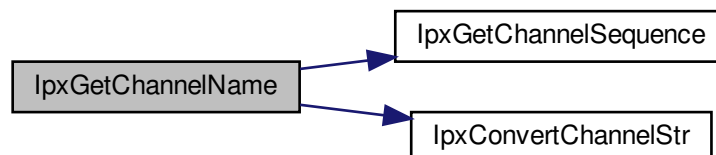
Parameters

<i>pixelType</i>	Pixel type.
<i>chnlIndx</i>	Channel index.

Returns

The return value is channel name.

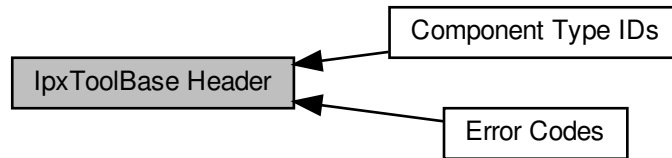
Here is the call graph for this function:



4.11 lpxToolBase Header

Macros, defines, structures for lpxToolBase and [lpxComponent](#) Class.

Collaboration diagram for lpxToolBase Header:



Modules

- [Error Codes](#)
Common Error Codes.
- [Component Type IDs](#)
Component Type IDs.

Classes

- struct [lpxRect](#)
The [lpxRect](#) structure defines a rectangle by the coordinates of its upper-left corner and width, height.
- struct [lpxSize](#)
The [lpxSize](#) structure specifies a rectangle.
- struct [lpxPoint](#)
The [lpxPoint](#) structure specifies a point.
- class [lpxComponent](#)
A Class for [lpxComponent](#) modules that contains methods for setting/getting/executing Component features.

Macros

- `#define IPX_ERR(__component__, __code__) ((__code__)==0) ? (IPX_ERR_OK) : ((0x80000000 | ((__component__)<<16) | (__code__)))`
Imperx Error Macro.
- `#define IPX_WRN(__component__, __code__) (lpxError)((__component__<<16) | __code__)`
Imperx Warning Macro.
- `#define IPX_ERR_SUCCEEDED(__code__) ((__code__ & 0xFFFF) ==0)`
Imperx Error Code Succeeded Macro.
- `#define IPX_ERR_FAILED(__code__) (__code__>0x80000000)`
Imperx Error Code Failed Macro.
- `#define IPX_ERR_WARNING(__code__) ((__code__<0x80000000) && (__code__!=0))`
Imperx Error Code Warning Macro.

Typedefs

- typedef void * [lpxHandle](#)
lpxHandle defines the handle of lpxTools component's instance.
- typedef uint32_t [lpxError](#)
Error definitions.

4.11.1 Detailed Description

Macros, defines, structures for lpxToolBase and [lpxComponent](#) Class.

4.12 Error Codes

Common Error Codes.

Collaboration diagram for Error Codes:



Macros

- `#define IPX_ERR_OK 0`
This error code occurs when the function was successful.
- `#define IPX_ERR_UNKNOWN 1`
This error code occurs when the function is not successful.
- `#define IPX_ERR_FILE_NOTFOUND 2`
This error code occurs when the file is not found.
- `#define IPX_ERR_NOT_SUPPORTED 3`
This error code occurs when the parameter uses or functionality is not supported.
- `#define IPX_ERR_ACCESS_DENIED 4`
This error code occurs when access is denied.
- `#define IPX_ERR_OUT_OF_RANGE 5`
This error code occurs when the parameter valid set is out of range.
- `#define IPX_ERR_BUFFER_TOO_SMALL 6`
This error code occurs when the buffer is too small.
- `#define IPX_ERR_INVALID_ARGUMENT 7`
This error code occurs when the argument passed in is an invalid argument.
- `#define IPX_ERR_NULL_POINTER 8`
This error code occurs when the parameter, source image, or destination image are unable to be created causing a null pointer.
- `#define IPX_ERR_NOT_ENOUGH_MEMORY 9`
This error code occurs when not enough memory was declared for the destination image.
- `#define IPX_ERR_NOT_IMPLEMENTED 10`
This error code occurs when the function, parameter, or feature has not been implemented.

4.12.1 Detailed Description

Common Error Codes.

This is the Common error codes

4.13 Component Type IDs

Component Type IDs.

Collaboration diagram for Component Type IDs:



Macros

- `#define IPX_CMP_IMG_SERIALIZER 0x01`
lpxSerializer Component Type.
- `#define IPX_CMP_BAYER_DEMOSAICING 0x05`
lpxDemosaic Component Type.
- `#define IPX_CMP_TS_DEMOSAICING 0x06`
lpxDemosaic Component Type.
- `#define IPX_CMP_DISPLAY 0x07`
lpxDisplay Component Type.
- `#define IPX_CMP_IMG_CONVERTER 0x08`
lpxImageConverter Component Type.
- `#define IPX_CMP_IMG_UNPACKER 0x09`
lpxImageUnacker Component Type.

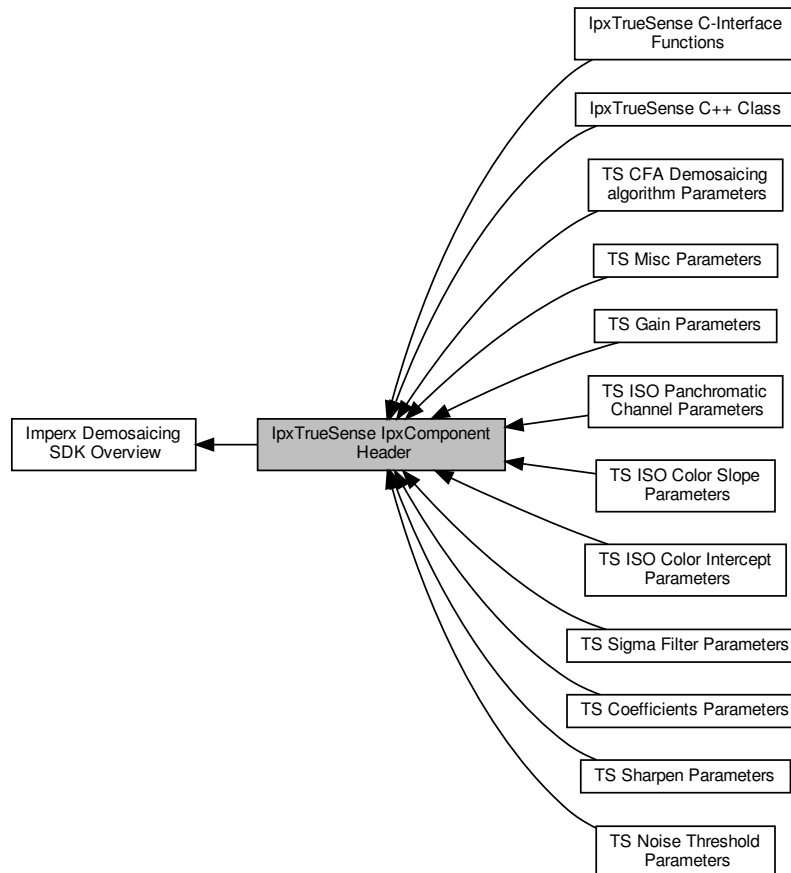
4.13.1 Detailed Description

Component Type IDs.

4.14 IpxTrueSense IpxComponent Header

TrueSense functions and classes with [IpxComponent](#) features.

Collaboration diagram for IpxTrueSense IpxComponent Header:



Modules

- [TS CFA Demosaicing algorithm Parameters](#)
Defines for TS CFA Demosaicing algorithms.
- [TS Misc Parameters](#)
Defines for TS Misc parameters.
- [TS Gain Parameters](#)
Defines for TS gain parameters.
- [TS ISO Panchromatic Channel Parameters](#)
Defines for TS ISO Panchromatic channel parameters.
- [TS ISO Color Slope Parameters](#)

- Defines for TS ISO Color Slope parameters.*
- [TS ISO Color Intercept Parameters](#)
Defines for TS ISO Color Intercept parameters.
- [TS Sigma Filter Parameters](#)
Defines for TS Sigma Filter parameters.
- [TS Coefficients Parameters](#)
Defines for TS Coefficients parameters.
- [TS Sharpen Parameters](#)
Defines for TS Sharpen parameters.
- [TS Noise Threshold Parameters](#)
Defines for TS Noise Threshold parameters.
- [lpxTrueSense C++ Class](#)
C++ Class for `lpxTrueSense`.
- [lpxTrueSense C-Interface Functions](#)
C-interface functions for `lpxTrueSense`.

4.14.1 Detailed Description

TrueSense functions and classes with `lpxComponent` features.

This module is responsible for conversion CFA pattern (TRUESENSE) to color image.

4.15 IpxUserData Header

Defines for user data types for Images.

Classes

- struct [IpxUserData](#)
Data structure for description of User Data linked with Imperx Image.

Enumerations

- enum [IPX_USER_DATA](#) : unsigned long { [IPX_NOT_DATA](#) = 0, [IPX_HASHTABLE_DATA](#), [IPX_XML_DATA](#), [IPX_CUSTOM_DATA](#) }
Definition of user data types for Images which are processed.

4.15.1 Detailed Description

Defines for user data types for Images.

4.15.2 Enumeration Type Documentation

4.15.2.1 IPX_USER_DATA

```
enum IPX\_USER\_DATA : unsigned long
```

Definition of user data types for Images which are processed.

Note

The User data are intended to store additional information about the image

See also

[IpxUserData](#)
[IpxCreateUserData](#)
[IpxReleaseUserData](#)

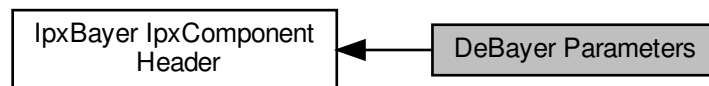
Enumerator

IPX_NOT_DATA	Type of user data is undefined.
IPX_HASHTABLE_DATA	User data are placed into hashtable.
IPX_XML_DATA	User data have XML format.
IPX_CUSTOM_DATA	Format of user data is defined by customer.

4.16 DeBayer Parameters

Defines for DeBayer Parameters.

Collaboration diagram for DeBayer Parameters:



Macros

- `#define DEBAYER_ALGO_TYPE "BayerAlgType"`
- `#define DEBAYER_NOREALLOCT "NoRealloc"`

4.16.1 Detailed Description

Defines for DeBayer Parameters.

Table 4.41 DeBayer Parameters

Macro	Parameter Name	Type and Range	Description
DEBAYER_ALGO_TYPE	"BayerAlgType"	[int: 0,4]	Bayer Algorithm Type
DEBAYER_NOREALLOCT	"NoRealloc"	[int: 0,1]	No Realloc enabled

4.16.2 Macro Definition Documentation

4.16.2.1 DEBAYER_ALGO_TYPE

```
#define DEBAYER_ALGO_TYPE "BayerAlgType"
```

Bayer Algorithm Type

Type/Range [int: 0,4]

Note

Used by SetParamInt and GetParamInt

4.16.2.2 DEBAYER_NOREALLOC

```
#define DEBAYER_NOREALLOC "NoRealloc"
```

No Realloc enabled

Type/Range [int: 0,1]

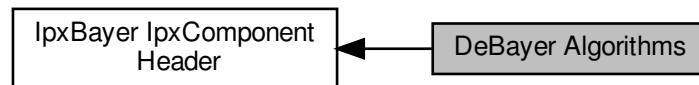
Note

Used by SetParamInt and GetParamInt

4.17 DeBayer Algorithms

Type of DeBayer Algorithms.

Collaboration diagram for DeBayer Algorithms:



Macros

- `#define BAYER_SIMPLE 0`
- `#define BAYER_GRADIENT 1`
- `#define BAYER_EA 2`
- `#define BAYER_OPENGL_MHC 3`
- `#define BAYER_OPENGL_MMA 4`

4.17.1 Detailed Description

Type of DeBayer Algorithms.

Defines DeBayer Algorithms

Note

These parameters are used in the SetIntParam function to program the DEBAYER_ALGO_TYPE parameter

For example,

```
pDeBayer->GetComponent()->SetParamInt(DEBAYER_ALGO_TYPE, BAYER_AHD);
```

4.17.2 Macro Definition Documentation

4.17.2.1 BAYER_SIMPLE

```
#define BAYER_SIMPLE 0
```

Simple algorithm. Average quality, high speed.

4.17.2.2 BAYER_GRADIENT

```
#define BAYER_GRADIENT 1
```

Gradient Based algorithm. High quality, medium speed.

4.17.2.3 BAYER_EA

```
#define BAYER_EA 2
```

Edge-Aware Demosaicing. Average quality, medium speed.

4.17.2.4 BAYER_OPENGL_MHC

```
#define BAYER_OPENGL_MHC 3
```

OpenGL MHC Algorithm.

4.17.2.5 BAYER_OPENGL_MMA

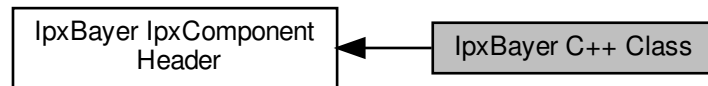
```
#define BAYER_OPENGL_MMA 4
```

OpenGL MMA Algorithm.

4.18 IpxBayer C++ Class

C++ Class for [IpxBayer](#).

Collaboration diagram for IpxBayer C++ Class:



Classes

- class [IpxBayer](#)

A Class for [IpxBayer](#) modules that contains methods to convert Bayer CFA (Color Filter Array) images.

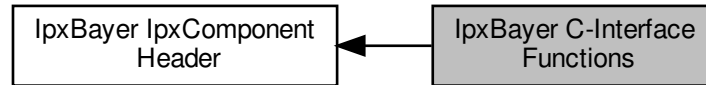
4.18.1 Detailed Description

C++ Class for [IpxBayer](#).

4.19 lpxBayer C-Interface Functions

C-interface functions for [lpxBayer](#).

Collaboration diagram for lpxBayer C-Interface Functions:



Functions

- BAYER_EXTERN_C BAYER_API [lpxHandle](#) BAYER_CALL [lpxBayer_CreateComponent](#) ()
This C-interface function returns the [lpxHandle](#) for the created [lpxBayer](#) instance.
- BAYER_EXTERN_C BAYER_API void BAYER_CALL [lpxBayer_DeleteComponent](#) ([lpxHandle](#) hBayer)
This C-interface function deletes the [lpxHandle](#) hBayer component and all associated resources obtained by the [lpxBayer](#) object.
- BAYER_EXTERN_C BAYER_API [lpxHandle](#) BAYER_CALL [lpxBayer_GetComponent](#) ([lpxHandle](#) hBayer)
This C-interface function returns the [lpxHandle](#) for the created [lpxBayer](#) instance.
- BAYER_EXTERN_C BAYER_API [lpxError](#) BAYER_CALL [lpxBayer_ConvertImage](#) ([lpxHandle](#) hBayer, const [lpxImage](#) *pSrc, [lpxImage](#) *pDst)
This C-interface function converts the input source [lpxImage](#) to the targeted output destination.
- BAYER_EXTERN_C BAYER_API [lpxError](#) BAYER_CALL [lpxBayer_AllocData](#) ([lpxHandle](#) hBayer, const [lpxImage](#) *pSrc, [lpxImage](#) *pDst)
This C-interface function allocates the data.
- BAYER_EXTERN_C BAYER_API void BAYER_CALL [lpxBayer_ReleaseData](#) ([lpxHandle](#) hBayer)
This C-interface function release the [lpxHandle](#) to the [lpxBayer](#) data.

4.19.1 Detailed Description

C-interface functions for [lpxBayer](#).

4.19.2 Function Documentation

4.19.2.1 IpxBayer_CreateComponent()

```
BAYER_EXTERN_C BAYER_API IpxHandle BAYER_CALL IpxBayer_CreateComponent ( )
```

This C-interface function returns the IpxHandle for the created IpxBayer instance.

Returns

Returns the IpxHandle for the created IpxBayer object

4.19.2.2 IpxBayer_DeleteComponent()

```
BAYER_EXTERN_C BAYER_API void BAYER_CALL IpxBayer_DeleteComponent (
    IpxHandle hBayer )
```

This C-interface function deletes the IpxHandle hBayer component and all associated resources obtained by the IpxBayer object.

Parameters

in	hBayer	Pointer to the IpxHandle for the IpxBayer instance
----	--------	--

Returns

void

4.19.2.3 IpxBayer_GetComponent()

```
BAYER_EXTERN_C BAYER_API IpxHandle BAYER_CALL IpxBayer_GetComponent (
    IpxHandle hBayer )
```

This C-interface function returns the IpxHandle for the created IpxBayer instance.

Parameters

in	hBayer	Pointer to the IpxHandle for the IpxBayer instance
----	--------	--

Returns

Returns the IpxHandle for the IpxBayer object component

4.19.2.4 IpxBayer_ConvertImage()

```
BAYER_EXTERN_C BAYER_API IpxError BAYER_CALL IpxBayer_ConvertImage (
    IpxHandle hBayer,
    const IpxImage * pSrc,
    IpxImage * pDst )
```

This C-interface function converts the input source [IpxImage](#) to the targeted output destination.

Parameters

in	<i>hBayer</i>	Pointer to the IpxHandle for the IpxBayer Component
in	<i>pSrc</i>	Pointer to the source IpxImage
out	<i>pDst</i>	Pointer to the output destination IpxImage

Returns

Returns the error code:

- [IPX_ERR_OK](#) Successfully converts the source [IpxImage](#) to the targeted output destination [IpxImage](#)
- If IpxError error code < 0, then it returns a negative error code indicating problems converting the [IpxImage](#)

4.19.2.5 IpxBayer_AllocData()

```
BAYER_EXTERN_C BAYER_API IpxError BAYER_CALL IpxBayer_AllocData (
    IpxHandle hBayer,
    const IpxImage * pSrc,
    IpxImage * pDst )
```

This C-interface function allocates the data.

Parameters

in	<i>hBayer</i>	Pointer of the IpxHandle for the IpxBayer Component
in	<i>pSrc</i>	Pointer to the source IpxImage
in	<i>pDst</i>	Pointer to the output destination IpxImage

Returns

Returns the error code:

- [IPX_ERR_OK](#) Successfully allocates the [IpxImage](#) data.
- If IpxError error code < 0, then it returns a negative error code indicating problems allocating [IpxImage](#) data

4.19.2.6 IpxBayer_ReleaseData()

```
BAYER_EXTERN_C BAYER_API void BAYER_CALL IpxBayer_ReleaseData (
    IpxHandle hBayer )
```

This C-interface function release the IpxHandle to the IpxBayer data.

Parameters

in	<i>hBayer</i>	Pointer of the IpxHandle for the IpxBayer data
----	---------------	--

Returns

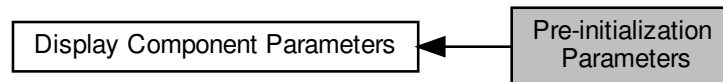
Returns the error code:

- `IPX_ERR_OK` Successfully releases the IpxBayer data.
- If IpxError error code < 0, then it returns a negative error code indicating problems releasing the IpxBayer data

4.20 Pre-initialization Parameters

Defines for Pre-Initialization Parameters.

Collaboration diagram for Pre-initialization Parameters:



Macros

- `#define IDP_BACKGROUND "display.bg.color"`
- `#define IDP_INIT_FIT "display.init.fit"`
- `#define IDP_INIT_AT_X "display.init.at.x"`
- `#define IDP_INIT_AT_Y "display.init.at.y"`
- `#define IDP_SMOOTHING "processing.smoothing"`
- `#define IDP_OGL_BAYER "ogl.processing.bayer.method"`
- `#define IDP_OGL_TRUESENSE "ogl.processing.truesense.method"`
- `#define IDP_GDI_BAYER "gdi.processing.bayer.method"`
- `#define IDP_GDI_TRUESENSE "gdi.processing.truesense.method"`
- `#define IDP_COMMAND_WINDOW "window.command"`
- `#define IDP_OVERLAY_FONT_DESC_0 "overlay.font.desc.0"`
- `#define IDP_OVERLAY_FONT_DESC_1 "overlay.font.desc.1"`
- `#define IDP_OVERLAY_FONT_DESC_2 "overlay.font.desc.2"`
- `#define IDP_OVERLAY_FONT_DESC_3 "overlay.font.desc.3"`

4.20.1 Detailed Description

Defines for Pre-Initialization Parameters.

Table 4.47 PRE-INIT PARAMETERS

Macro	Parameter Name	Type	Description
IDP_BACKGROUND	"display.bg.color"	[int: 0, 1]	Background color
IDP_SMOOTHING	"processing.smoothing"	[int: 0,1]	Smoothing
IDP_OGL_BAYER	"ogl.processing.bayer.method"	[int: 0,1]	De-bayer method for OpenGL mode
IDP_OGL_TRUESENSE	"ogl.processing.truesense.↵ method"	[int: 0,1]	Truesense demosaicing method for OpenGL mode
IDP_GDI_BAYER	"gdi.processing.bayer.method"	[int: 0-2]	De-bayer method for GDI mode

Macro	Parameter Name	Type	Description
IDP_GDI_TRUESENSE	"gdi.processing.truesense.↔ method"	[int: 0,1]	Truesense demosaicing method for GDI mode
IDP_COMMAND_WINDOW	"window.command"	[int]	Command window handle
IDP_OVERLAY_FONT_DESCRIPTOR_0	"overlay.font.desc.0"	[char*]	Overlay font descriptor for font #0
IDP_OVERLAY_FONT_DESCRIPTOR_1	"overlay.font.desc.1"	[char*]	Overlay font descriptor for font #1
IDP_OVERLAY_FONT_DESCRIPTOR_2	"overlay.font.desc.2"	[char*]	Overlay font descriptor for font #2
IDP_OVERLAY_FONT_DESCRIPTOR_3	"overlay.font.desc.3"	[char*]	Overlay font descriptor for font #3

4.20.2 Macro Definition Documentation

4.20.2.1 IDP_BACKGROUND

```
#define IDP_BACKGROUND "display.bg.color"
```

Background color

Type/Range [int: 0, 1]

Note

Used by SetParamInt and GetParamInt [QT: yes]

4.20.2.2 IDP_INIT_FIT

```
#define IDP_INIT_FIT "display.init.fit"
```

View: default fit mode for new image format

Type/Range [int: 0-3]

Note

Used by SetParamInt and GetParamInt [QT: no]

4.20.2.3 IDP_INIT_AT_X

```
#define IDP_INIT_AT_X "display.init.at.x"
```

View: default fit mode for new image format

Type/Range [int: 0-3]

Note

Used by SetParamInt and GetParamInt [QT: no]

4.20.2.4 IDP_INIT_AT_Y

```
#define IDP_INIT_AT_Y "display.init.at.y"
```

View: default fit mode for new image format

Type/Range [int: 0-3]

Note

Used by SetParamInt and GetParamInt [QT: no]

4.20.2.5 IDP_SMOOTHING

```
#define IDP_SMOOTHING "processing.smoothing"
```

Smoothing

Type/Range [int: 0,1]

Note

Used by SetParamInt and GetParamInt [QT: no]

4.20.2.6 IDP_OGL_BAYER

```
#define IDP_OGL_BAYER "ogl.processing.bayer.method"
```

De-bayer method for OpenGL mode

Type/Range [int: 0,1]

Note

Used by SetParamInt and GetParamInt [QT: no]

4.20.2.7 IDP_OGL_TRUESENSE

```
#define IDP_OGL_TRUESENSE "ogl.processing.truesense.method"
```

Truesense demosaicing method for OpenGL mode

Type/Range [int: 0,1]

Note

Used by SetParamInt and GetParamInt [QT: no]

4.20.2.8 IDP_GDI_BAYER

```
#define IDP_GDI_BAYER "gdi.processing.bayer.method"
```

De-bayer method for GDI mode

Type/Range [int: 0-2]

Note

Used by SetParamInt and GetParamInt [QT: no]

4.20.2.9 IDP_GDI_TRUESENSE

```
#define IDP_GDI_TRUESENSE "gdi.processing.truesense.method"
```

Truesense demosaicing method for GDI mode

Type/Range [int: 0,1]

Note

Used by SetParamInt and GetParamInt [QT: no]

4.20.2.10 IDP_COMMAND_WINDOW

```
#define IDP_COMMAND_WINDOW "window.command"
```

Command window handle

Type/Range [int]

Note

Used by SetParamInt and GetParamInt [QT: yes]

4.20.2.11 IDP_OVERLAY_FONT_DESC_0

```
#define IDP_OVERLAY_FONT_DESC_0 "overlay.font.desc.0"
```

Overlay font descriptor for font #0

Type/Range [char*]

Note

Used by SetParamString and GetParamString [QT: no]

4.20.2.12 IDP_OVERLAY_FONT_DESC_1

```
#define IDP_OVERLAY_FONT_DESC_1 "overlay.font.desc.1"
```

Overlay font descriptor for font #1

Type/Range [char*]

Note

Used by SetParamString and GetParamString [QT: no]

4.20.2.13 IDP_OVERLAY_FONT_DESC_2

```
#define IDP_OVERLAY_FONT_DESC_2 "overlay.font.desc.2"
```

Overlay font descriptor for font #2

Type/Range [char*]

Note

Used by SetParamString and GetParamString [QT: no]

4.20.2.14 IDP_OVERLAY_FONT_DESC_3

```
#define IDP_OVERLAY_FONT_DESC_3 "overlay.font.desc.3"
```

Overlay font descriptor for font #3

Type/Range [char*]

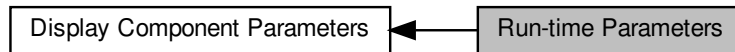
Note

Used by SetParamString and GetParamString [QT: no]

4.21 Run-time Parameters

Defines for Run-time Parameters (View Management)

Collaboration diagram for Run-time Parameters:



Macros

- `#define IDP_SIGNATURE "system.signature"`
- `#define IDP_VIEW_FIT "display.fit"`
- `#define IDP_VIEW_X "display.x"`
- `#define IDP_VIEW_Y "display.y"`
- `#define IDP_VIEW_SCALE "display.view.scale"`
- `#define IDP_MANAGED_FPS "playback.managed.fps"`
- `#define IDP_MANAGED_STATE "playback.managed"`
- `#define IDP_VIEW_CLR "display.view.color"`
- `#define IDP_VIEW_CURSOR_X "display.view.cursor.x"`
- `#define IDP_VIEW_CURSOR_Y "display.view.cursor.y"`
- `#define IDP_PROC_PROCESSOR "processor"`
- `#define IDP_PROC_PROCESSOR_TYPE "processor.type"`
- `#define IDP_MENU_X "menu.x"`
- `#define IDP_MENU_Y "menu.y"`
- `#define IDP_MENU_CMD "menu.cmd"`

4.21.1 Detailed Description

Defines for Run-time Parameters (View Management)

Table 4.48 RUN-TIME PARAMETERS

Macro	Parameter Name	Type	Description
IDP_VIEW_FIT	"display.fit"	[int: 0-3]	View: current fit mode
IDP_VIEW_X	"display.x"	[real: 0-1]	X position
IDP_VIEW_Y	"display.y"	[real: 0-1]	Y position
IDP_MANAGED_FPS	"playback.managed.fps"	[int]	"Managed" state FPS (default 20)
IDP_MANAGED_STATE	"playback.managed"	[int]	"Managed" state flag (default 0)

4.21.2 Macro Definition Documentation

4.21.2.1 IDP_SIGNATURE

```
#define IDP_SIGNATURE "system.signature"
```

Component identifier [QT: no]

4.21.2.2 IDP_VIEW_FIT

```
#define IDP_VIEW_FIT "display.fit"
```

View: current fit mode

Type/Range [int: 0-3]

Note

Used by SetParamInt and GetParamInt [QT: yes]

4.21.2.3 IDP_VIEW_X

```
#define IDP_VIEW_X "display.x"
```

X position

Type/Range [real: 0-1] [QT: no]

4.21.2.4 IDP_VIEW_Y

```
#define IDP_VIEW_Y "display.y"
```

Y position

Type/Range [real: 0-1] [QT: no]

4.21.2.5 IDP_VIEW_SCALE

```
#define IDP_VIEW_SCALE "display.view.scale"
```

View: current scale [QT: no]

4.21.2.6 IDP_MANAGED_FPS

```
#define IDP_MANAGED_FPS "playback.managed.fps"
```

"Managed" state FPS (default 20)

Type/Range [int] [QT: no]

4.21.2.7 IDP_MANAGED_STATE

```
#define IDP_MANAGED_STATE "playback.managed"
```

"Managed" state flag (default 0)

Type/Range [int] [QT: no]

4.21.2.8 IDP_VIEW_CLR

```
#define IDP_VIEW_CLR "display.view.color"
```

Current color

Type/Range [char*] [QT: no]

4.21.2.9 IDP_VIEW_CURSOR_X

```
#define IDP_VIEW_CURSOR_X "display.view.cursor.x"
```

Current cursor X position in image co-ordinates

Type/Range [int: 0-width] [QT: no]

4.21.2.10 IDP_VIEW_CURSOR_Y

```
#define IDP_VIEW_CURSOR_Y "display.view.cursor.y"
```

Current cursor Y position in image co-ordinates

Type/Range [int: 0-width] [QT: no]

4.21.2.11 IDP_PROC_PROCESSOR

```
#define IDP_PROC_PROCESSOR "processor"
```

Image processor pointer [QT: no]

4.21.2.12 IDP_PROC_PROCESSOR_TYPE

```
#define IDP_PROC_PROCESSOR_TYPE "processor.type"
```

Image processor type [QT: no]

4.21.2.13 IDP_MENU_X

```
#define IDP_MENU_X "menu.x"
```

X position for context menu

Type/Range [int] [QT: no]

4.21.2.14 IDP_MENU_Y

```
#define IDP_MENU_Y "menu.y"
```

Y position for context menu

Type/Range [int] [QT: no]

4.21.2.15 IDP_MENU_CMD

```
#define IDP_MENU_CMD "menu.cmd"
```

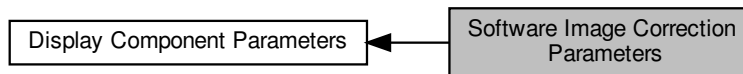
Context menu command

Type/Range [int] [QT: no]

4.22 Software Image Correction Parameters

Defines for Software Image Correction Parameters.

Collaboration diagram for Software Image Correction Parameters:



Macros

- #define `IDP_CORR_MODE` "correction.mode"
- #define `IDP_CORR_GAIN_R` "correction.gain.r"
- #define `IDP_CORR_GAIN_G` "correction.gain.g"
- #define `IDP_CORR_GAIN_B` "correction.gain.b"
- #define `IDP_CORR_OFFS_R` "correction.off.s.r"
- #define `IDP_CORR_OFFS_G` "correction.off.s.g"
- #define `IDP_CORR_OFFS_B` "correction.off.s.b"
- #define `IDP_CORR_GAMMA` "correction.gamma"

4.22.1 Detailed Description

Defines for Software Image Correction Parameters.

Table 4.49 SW IMAGE CORRECTION PARAMETERS

Macro	Parameter Name	Type	Description
<code>IDP_CORR_MODE</code>	"correction.mode"	[int: 0-2]	Software Correction: current mode
<code>IDP_CORR_GAIN_R</code>	"correction.gain.r"	[real: 0+]	Software Correction: Gain for red channel
<code>IDP_CORR_GAIN_G</code>	"correction.gain.g"	[real: 0+]	Software Correction: Gain for green channel
<code>IDP_CORR_GAIN_B</code>	"correction.gain.b"	[real: 0+]	Software Correction: Gain for blue channel
<code>IDP_CORR_OFFS_R</code>	"correction.off.s.r"	[int]	Software Correction: Offset for red channel
<code>IDP_CORR_OFFS_G</code>	"correction.off.s.g"	[int]	Software Correction: Offset for green channel
<code>IDP_CORR_OFFS_B</code>	"correction.off.s.b"	[int]	Software Correction: Offset for blue channel
<code>IDP_CORR_GAMMA</code>	"correction.gamma"	[real: 0+]	Software Correction: Gamma

4.22.2 Macro Definition Documentation

4.22.2.1 IDP_CORR_MODE

```
#define IDP_CORR_MODE "correction.mode"
```

Software Correction: current mode

Type/Range [int: 0-2]

Note

Used by SetParamInt and GetParamInt [QT: no]

4.22.2.2 IDP_CORR_GAIN_R

```
#define IDP_CORR_GAIN_R "correction.gain.r"
```

Software Correction: Gain for red channel

Type/Range [real: 0+]

Note

Used by SetParamFloat and GetParamFloat [QT: no]

4.22.2.3 IDP_CORR_GAIN_G

```
#define IDP_CORR_GAIN_G "correction.gain.g"
```

Software Correction: Gain for green channel

Type/Range [real: 0+]

Note

Used by SetParamFloat and GetParamFloat [QT: no]

4.22.2.4 IDP_CORR_GAIN_B

```
#define IDP_CORR_GAIN_B "correction.gain.b"
```

Software Correction: Gain for blue channel

Type/Range [real: 0+]

Note

Used by SetParamFloat and GetParamFloat [QT: no]

4.22.2.5 IDP_CORR_OFFS_R

```
#define IDP_CORR_OFFS_R "correction.off.s.r"
```

Software Correction: Offset for red channel

Type/Range [int]

Note

Used by SetParamInt and GetParamInt [QT: no]

4.22.2.6 IDP_CORR_OFFS_G

```
#define IDP_CORR_OFFS_G "correction.off.s.g"
```

Software Correction: Offset for green channel

Type/Range [int]

Note

Used by SetParamInt and GetParamInt [QT: no]

4.22.2.7 IDP_CORR_OFFS_B

```
#define IDP_CORR_OFFS_B "correction.offfs.b"
```

Software Correction: Offset for blue channel

Type/Range [int]

Note

Used by SetParamInt and GetParamInt [QT: no]

4.22.2.8 IDP_CORR_GAMMA

```
#define IDP_CORR_GAMMA "correction.gamma"
```

Software Correction: Gamma

Type/Range [real: 0+]

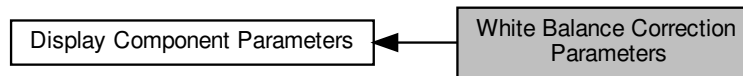
Note

Used by SetParamFloat and GetParamFloat [QT: no]

4.23 White Balance Correction Parameters

Defines for White Balance Correction Parameters.

Collaboration diagram for White Balance Correction Parameters:



Macros

- `#define IDP_CALC_COEF_R "correction.calc.r"`
- `#define IDP_CALC_COEF_G "correction.calc.g"`
- `#define IDP_CALC_COEF_B "correction.calc.b"`

4.23.1 Detailed Description

Defines for White Balance Correction Parameters.

Table 4.50 WHITE BALANCE CORRECTION PARAMETERS

Macro	Parameter Name	Type	Description
IDP_CALC_COEF_R	"correction.calc.r"	[real: 0+]	White balance: coef for red channel
IDP_CALC_COEF_G	"correction.calc.g"	[real: 0+]	White balance: coef for green channel
IDP_CALC_COEF_B	"correction.calc.b"	[real: 0+]	White balance: coef for blue channel

4.23.2 Macro Definition Documentation

4.23.2.1 IDP_CALC_COEF_R

```
#define IDP_CALC_COEF_R "correction.calc.r"
```

White balance: coef for red channel

Type/Range [real: 0+]

Note

Used by SetParamFloat and GetParamFloat [QT: no]

4.23.2.2 IDP_CALC_COEF_G

```
#define IDP_CALC_COEF_G "correction.calc.g"
```

White balance: coef for green channel

Type/Range [real: 0+]

Note

Used by SetParamFloat and GetParamFloat [QT: no]

4.23.2.3 IDP_CALC_COEF_B

```
#define IDP_CALC_COEF_B "correction.calc.b"
```

White balance: coef for blue channel

Type/Range [real: 0+]

Note

Used by SetParamFloat and GetParamFloat [QT: no]

4.24 Overlay Text Parameters

Defines for Overlay Text Parameters.

Collaboration diagram for Overlay Text Parameters:



Macros

- `#define IDP_OVERLAY_INDEX "overlay.index"`
- `#define IDP_OVERLAY_POS "overlay.pos"`
- `#define IDP_OVERLAY_FONT "overlay.font"`
- `#define IDP_OVERLAY_COLOR "overlay.clr"`
- `#define IDP_OVERLAY_BGMODE "overlay.bgmode"`
- `#define IDP_OVERLAY_TEXT "overlay.text"`

4.24.1 Detailed Description

Defines for Overlay Text Parameters.

Table 4.51 OVERLAY TEXT PARAMETERS

Macro	Parameter Name	Type	Description
IDP_OVERLAY_INDEX	"overlay.index"	[int: 0-3]	Overlay: current index
IDP_OVERLAY_POS	"overlay.pos"	[int: 0-8]	Overlay: position
IDP_OVERLAY_FONT	"overlay.font"	[int: 0-3]	Overlay: font index
IDP_OVERLAY_COLOR	"overlay.clr"	[int]	Overlay: text color
IDP_OVERLAY_BGMODE	"overlay.bgmode"	[int: 0-3]	Overlay: background mode
IDP_OVERLAY_TEXT	"overlay.text"	[char*]	Overlay: text

4.24.2 Macro Definition Documentation

4.24.2.1 IDP_OVERLAY_INDEX

```
#define IDP_OVERLAY_INDEX "overlay.index"
```

Overlay: current index

Type/Range [int: 0-3]

Note

Used by SetParamInt and GetParamInt [QT: no]

4.24.2.2 IDP_OVERLAY_POS

```
#define IDP_OVERLAY_POS "overlay.pos"
```

Overlay: position

Type/Range [int: 0-8]

Note

Used by SetParamInt and GetParamInt [QT: no]

4.24.2.3 IDP_OVERLAY_FONT

```
#define IDP_OVERLAY_FONT "overlay.font"
```

Overlay: font index

Type/Range [int: 0-3]

Note

Used by SetParamInt and GetParamInt [QT: no]

4.24.2.4 IDP_OVERLAY_COLOR

```
#define IDP_OVERLAY_COLOR "overlay.clr"
```

Overlay: text color

Type/Range [int]

Note

Used by SetParamInt and GetParamInt [QT: no]

4.24.2.5 IDP_OVERLAY_BGMODE

```
#define IDP_OVERLAY_BGMODE "overlay.bgmode"
```

Overlay: background mode

Type/Range [int: 0-3]

Note

Used by SetParamInt and GetParamInt [QT: no]

4.24.2.6 IDP_OVERLAY_TEXT

```
#define IDP_OVERLAY_TEXT "overlay.text"
```

Overlay: text

Type/Range [char*]

Note

Used by SetParamString and GetParamString [QT: no]

4.25 Dump Rect Parameters

Defines for Dump Rect Parameters.

Collaboration diagram for Dump Rect Parameters:



Macros

- `#define IDP_DUMP_X "dump.x"`
- `#define IDP_DUMP_Y "dump.y"`
- `#define IDP_DUMP_W "dump.w"`
- `#define IDP_DUMP_H "dump.h"`
- `#define IDP_DUMP_COLOR "dump.clr"`

4.25.1 Detailed Description

Defines for Dump Rect Parameters.

Table 4.52 DUMP RECT PARAMETERS

Macro	Parameter Name	Type	Description
IDP_DUMP_X	"dump.x"	[int: 1-w]	Dump rect: x-pos
IDP_DUMP_Y	"dump.y"	[int: 1-h]	Dump rect: y-pos
IDP_DUMP_W	"dump.w"	[int: 1-(w-x)]	Dump rect: width
IDP_DUMP_H	"dump.h"	[int: 1-(h-y)]	Dump rect: height
IDP_DUMP_COLOR	"dump.clr"	[int]	Dump rect: color (optional)

4.25.2 Macro Definition Documentation

4.25.2.1 IDP_DUMP_X

```
#define IDP_DUMP_X "dump.x"
```

Dump rect: x-pos

Type/Range [int: 1-w]

Note

Used by SetParamInt and GetParamInt [QT: yes]

4.25.2.2 IDP_DUMP_Y

```
#define IDP_DUMP_Y "dump.y"
```

Dump rect: y-pos

Type/Range [int: 1-h]

Note

Used by SetParamInt and GetParamInt [QT: yes]

4.25.2.3 IDP_DUMP_W

```
#define IDP_DUMP_W "dump.w"
```

Dump rect: width

Type/Range [int: 1-(w-x)]

Note

Used by SetParamInt and GetParamInt [QT: yes]

4.25.2.4 IDP_DUMP_H

```
#define IDP_DUMP_H "dump.h"
```

Dump rect: height

Type/Range [int: 1-(h-y)]

Note

Used by SetParamInt and GetParamInt [QT: yes]

4.25.2.5 IDP_DUMP_COLOR

```
#define IDP_DUMP_COLOR "dump.clr"
```

Dump rect: color (optional)

Type/Range [int]

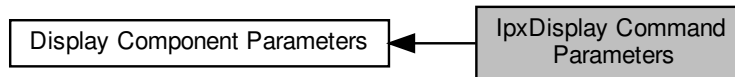
Note

Used by SetParamInt and GetParamInt [QT: no]

4.26 IpxDisplay Command Parameters

Defines for [IpxDisplay](#) Command Parameters.

Collaboration diagram for IpxDisplay Command Parameters:



Macros

- `#define IDPC_SET_CORRECTION` "processing.correction"
- `#define IDPC_CMD_VIEW_ZOOM_IN` "display.zoom.in"
- `#define IDPC_CMD_VIEW_ZOOM_OUT` "display.zoom.out"
- `#define IDPC_CMD_VIEW_ATCENTER` "display.atcenter"
- `#define IDPC_CMD_VIEW_AT` "display.center.at"
- `#define IDPC_CMD_VIEW_PARAMS` "display.params.set"
- `#define IDPC_CMD_CORR_CALC` "correction.calc"
- `#define IDPC_CMD_OVERLAY_SHOW` "overlay.show"
- `#define IDPC_CMD_OVERLAY_HIDE` "overlay.hide"
- `#define IDPC_CMD_MANAGED_ON` "playback.managed.on"
- `#define IDPC_CMD_MANAGED_OFF` "playback.managed.off"
- `#define IDPC_CMD_DUMP_ON` "display.dump.on"
- `#define IDPC_CMD_DUMP_OFF` "display.dump.off"
- `#define IDPC_CMD_FILTER_ADD` "filter.add"
- `#define IDPC_CMD_FILTER_DEL` "filter.del"
- `#define IDPC_CMD_PROC_ADD` "processing.proc.add"
- `#define IDPC_CMD_PROC_DEL` "processing.proc.del"
- `#define IDPC_CMD_MENU_SHOW` "display.menu.show"

4.26.1 Detailed Description

Defines for [IpxDisplay](#) Command Parameters.

Table 4.53 IPXDISPLAY COMMAND PARAMETERS

Macro	Parameter Name	Description
IDPC_SET_CORRECTION	"processing.correction"	Parameters IDP_CORR_XX) should be set before command call
IDPC_CMD_VIEW_ZOOM_IN	"display.zoom.in"	Zoom in (no params)
IDPC_CMD_VIEW_ZOOM_OUT	"display.zoom.out"	Zoom out (no params)

Macro	Parameter Name	Description
IDPC_CMD_VIEW_ATCENTER	"display.atcenter"	View at the center of image (no params)
IDPC_CMD_VIEW_AT	"display.center.at"	View at the specific position (IDP_VIEW_X, IDP_VIEW_Y should be set)
IDPC_CMD_CORR_CALC	"correction.calc"	Results placed in IDP_CALC_GAIN_XX parameters
IDPC_CMD_OVERLAY_SHOW	"overlay.show"	Show current overlay text with current parameters (current index specified by IDP_OVER_INDEX)
IDPC_CMD_OVERLAY_HIDE	"overlay.hide"	Hide current overlay text (current index specified by IDP_OVER_INDEX)
IDPC_CMD_MANAGED_ON	"playback.managed.on"	Set "managed" state (try to keep specified FPS value, see IDP_MANAGED_FPS)
IDPC_CMD_MANAGED_OFF	"playback.managed.off"	Clear "managed" state
IDPC_CMD_DUMP_ON	"display.dump.on"	Show dump rect (IDP_DUMP_X, IDP_DUMP_Y, IDP_DUMP_W, IDP_DUMP_H)
IDPC_CMD_DUMP_OFF	"display.dump.off"	Hide dump rect
IDPC_CMD_FILTER_ADD	"filter.add"	Add processing filter
IDPC_CMD_FILTER_DEL	"filter.del"	Remove processing filter

4.26.2 Macro Definition Documentation

4.26.2.1 IDPC_SET_CORRECTION

```
#define IDPC_SET_CORRECTION "processing.correction"
```

Parameters IDP_CORR_XX should be set before command call

Note

Used by RunCommand [QT: no]

4.26.2.2 IDPC_CMD_VIEW_ZOOM_IN

```
#define IDPC_CMD_VIEW_ZOOM_IN "display.zoom.in"
```

Zoom in (no params)

Note

Used by RunCommand [QT: yes]

4.26.2.3 IDPC_CMD_VIEW_ZOOM_OUT

```
#define IDPC_CMD_VIEW_ZOOM_OUT "display.zoom.out"
```

Zoom out (no params)

Note

Used by RunCommand [QT: yes]

4.26.2.4 IDPC_CMD_VIEW_ATCENTER

```
#define IDPC_CMD_VIEW_ATCENTER "display.atcenter"
```

View at the center of image (no params)

Note

Used by RunCommand [QT: yes]

4.26.2.5 IDPC_CMD_VIEW_AT

```
#define IDPC_CMD_VIEW_AT "display.center.at"
```

View at the specific position (IDP_VIEW_X, IDP_VIEW_Y should be set)

Note

Used by RunCommand [QT: no]

4.26.2.6 IDPC_CMD_VIEW_PARAMS

```
#define IDPC_CMD_VIEW_PARAMS "display.params.set"
```

Set View parameters

Note

Used by RunCommand [QT: no]

4.26.2.7 IDPC_CMD_CORR_CALC

```
#define IDPC_CMD_CORR_CALC "correction.calc"
```

Results placed in IDP_CALC_GAIN_XX parameters

Note

Used by RunCommand [QT: no]

4.26.2.8 IDPC_CMD_OVERLAY_SHOW

```
#define IDPC_CMD_OVERLAY_SHOW "overlay.show"
```

Show current overlay text with current parameters (current index specified by IDP_OVER_INDEX)

Note

Used by RunCommand [QT: no]

4.26.2.9 IDPC_CMD_OVERLAY_HIDE

```
#define IDPC_CMD_OVERLAY_HIDE "overlay.hide"
```

Hide current overlay text (current index specified by IDP_OVER_INDEX)

Note

Used by RunCommand [QT: no]

4.26.2.10 IDPC_CMD_MANAGED_ON

```
#define IDPC_CMD_MANAGED_ON "playback.managed.on"
```

Set "managed" state (try to keep specified FPS value, see IDP_MANAGED_FPS)

Note

Used by RunCommand [QT: no]

4.26.2.11 IDPC_CMD_MANAGED_OFF

```
#define IDPC_CMD_MANAGED_OFF "playback.managed.off"
```

Clear "managed" state

Note

Used by RunCommand [QT: no]

4.26.2.12 IDPC_CMD_DUMP_ON

```
#define IDPC_CMD_DUMP_ON "display.dump.on"
```

Show dump rect (IDP_DUMP_X, IDP_DUMP_Y, IDP_DUMP_W, IDP_DUMP_H)

Note

Used by RunCommand [QT: yes]

4.26.2.13 IDPC_CMD_DUMP_OFF

```
#define IDPC_CMD_DUMP_OFF "display.dump.off"
```

Hide dump rect

Note

Used by RunCommand [QT: yes]

4.26.2.14 IDPC_CMD_FILTER_ADD

```
#define IDPC_CMD_FILTER_ADD "filter.add"
```

Add processing filter [QT: no]

4.26.2.15 IDPC_CMD_FILTER_DEL

```
#define IDPC_CMD_FILTER_DEL "filter.del"
```

Remove processing filter [QT: no]

4.26.2.16 IDPC_CMD_PROC_ADD

```
#define IDPC_CMD_PROC_ADD "processing.proc.add"
```

Add processor [QT: no]

4.26.2.17 IDPC_CMD_PROC_DEL

```
#define IDPC_CMD_PROC_DEL "processing.proc.del"
```

Remove processor [QT: no]

4.26.2.18 IDPC_CMD_MENU_SHOW

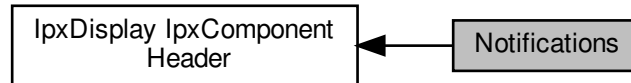
```
#define IDPC_CMD_MENU_SHOW "display.menu.show"
```

Show context menu at IDP_MENU_X, IDP_MENU_Y, result placed to IDP_MENU_CMD [QT: no]

4.27 Notifications

Defines for Notifications.

Collaboration diagram for Notifications:



Macros

- `#define IPXD_LBUTTON_DOWN 0x4002`
- `#define IPXD_LBUTTON_UP 0x4003`
- `#define IPXD_RBUTTON_DOWN 0x4004`
- `#define IPXD_CURSOR_MOVED 0x4008`
- `#define IPXD_KEY_DOWN 0x4009`
- `#define IPXD_VIEW_CHANGED 0x4010`
- `#define IPXD_CCLR_CHANGED 0x4012`
- `#define IPXD_PLAYBACK_FAILED 0x4014`
- `#define IPXD_ERROR_OPENGL 0x4300`

4.27.1 Detailed Description

Defines for Notifications.

4.27.2 Macro Definition Documentation

4.27.2.1 IPXD_LBUTTON_DOWN

```
#define IPXD_LBUTTON_DOWN 0x4002
```

Left mouse button down, param = MAKELONG(cursor.x, cursor.y), processing skipped if return = 1 [QT: yes]

4.27.2.2 IPXD_LBUTTON_UP

```
#define IPXD_LBUTTON_UP 0x4003
```

Left mouse button up, param = MAKELONG(cursor.x, cursor.y), processing skipped if return = 1 [QT: yes]

4.27.2.3 IPXD_RBUTTON_DOWN

```
#define IPXD_RBUTTON_DOWN 0x4004
```

Show context menu, param = MAKELONG(cursor.x, cursor.y) [QT: yes]

4.27.2.4 IPXD_CURSOR_MOVED

```
#define IPXD_CURSOR_MOVED 0x4008
```

Cursor moved, param = MAKELONG(cursor.x, cursor.y) [QT: yes]

4.27.2.5 IPXD_KEY_DOWN

```
#define IPXD_KEY_DOWN 0x4009
```

Keydown notification, param = MAKELONG(key code, repeat count) [QT: yes]

4.27.2.6 IPXD_VIEW_CHANGED

```
#define IPXD_VIEW_CHANGED 0x4010
```

View parameters changed [QT: yes]

4.27.2.7 IPXD_CCLR_CHANGED

```
#define IPXD_CCLR_CHANGED 0x4012
```

Current color changed [QT: yes]

4.27.2.8 IPXD_PLAYBACK_FAILED

```
#define IPXD_PLAYBACK_FAILED 0x4014
```

Playback failed (image format not supported), param 'reason' (IPXD_ERROR-x) [QT: mpno]

4.27.2.9 IPXD_ERROR_OPENGL

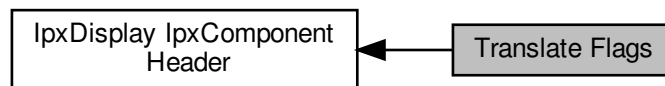
```
#define IPXD_ERROR_OPENGL 0x4300
```

Error rendering image with OpenGL, param = reason [QT: no]

4.28 Translate Flags

Defines for Translate Flags.

Collaboration diagram for Translate Flags:



Macros

- `#define IDFL_SCR_IMG 0x0001`
- `#define IDFL_IMG_SCR 0x0100`

4.28.1 Detailed Description

Defines for Translate Flags.

4.28.2 Macro Definition Documentation

4.28.2.1 IDFL_SCR_IMG

```
#define IDFL_SCR_IMG 0x0001
```

Translate flags: Screen to image [QT: yes]

4.28.2.2 IDFL_IMG_SCR

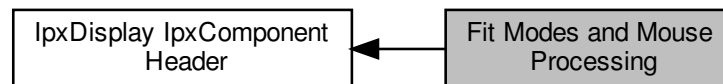
```
#define IDFL_IMG_SCR 0x0100
```

Translate flags: Image to screen [QT: yes]

4.29 Fit Modes and Mouse Processing

Defines for Fit Modes and Mouse Processing.

Collaboration diagram for Fit Modes and Mouse Processing:



Macros

- `#define IPXD_FIT_NONE 0L`
- `#define IPXD_FIT_WINDOW 1L`
- `#define IPXD_FIT_FILL 2L`
- `#define IPXD_FIT_FULLSIZE 3L`
- `#define IPXD_MOUSE_DEFAULT 0`
- `#define IPXD_MOUSE_SKIP 1`
- `#define IPXD_MOUSE_LOCK 2`

4.29.1 Detailed Description

Defines for Fit Modes and Mouse Processing.

4.29.2 Macro Definition Documentation

4.29.2.1 IPXD_FIT_NONE

```
#define IPXD_FIT_NONE 0L
```

Off [QT: yes]

4.29.2.2 IPXD_FIT_WINDOW

```
#define IPXD_FIT_WINDOW 1L
```

Fit to window [QT: yes]

4.29.2.3 IPXD_FIT_FILL

```
#define IPXD_FIT_FILL 2L
```

Fill window with image [QT: yes]

4.29.2.4 IPXD_FIT_FULLSIZE

```
#define IPXD_FIT_FULLSIZE 3L
```

Original size (100%) [QT: yes]

4.29.2.5 IPXD_MOUSE_DEFAULT

```
#define IPXD_MOUSE_DEFAULT 0
```

Do default processing [QT: no]

4.29.2.6 IPXD_MOUSE_SKIP

```
#define IPXD_MOUSE_SKIP 1
```

Skip mouse action [QT: no]

4.29.2.7 IPXD_MOUSE_LOCK

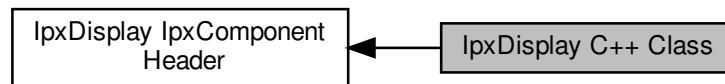
```
#define IPXD_MOUSE_LOCK 2
```

Lock mouse (capture) [QT: no]

4.30 IpxDisplay C++ Class

C++ Class for [IpxDisplay](#).

Collaboration diagram for IpxDisplay C++ Class:



Classes

- class [IpxDisplay](#)

A Class for [IpxDisplay](#) modules that contains methods to display [IpxImage](#) images. This class is responsible for displaying video frames and still images.

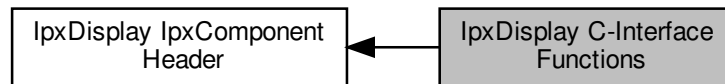
4.30.1 Detailed Description

C++ Class for [IpxDisplay](#).

4.31 lpxDisplay C-Interface Functions

C-interface functions for [lpxDisplay](#).

Collaboration diagram for lpxDisplay C-Interface Functions:



Functions

- IPXD_EXTERN_C IPXD_API [lpxHandle](#) IPXD_CALL [lpxDisplay_CreateComponent](#) ()
This C-interface function returns the [lpxHandle](#) for the created [lpxDisplay](#) instance.
- IPXD_EXTERN_C IPXD_API void IPXD_CALL [lpxDisplay_DeleteComponent](#) ([lpxHandle](#) hDisplay)
This C-interface function deletes the [lpxHandle](#) hDisplay component and all associated resources obtained by the [lpxDisplay](#) object.
- IPXD_EXTERN_C IPXD_API [lpxComponent](#) *IPXD_CALL [lpxDisplay_GetComponent](#) ([lpxHandle](#) hDisplay)
This C-interface function returns the [lpxHandle](#) for the created [lpxImageConverter](#) instance.
- IPXD_EXTERN_C IPXD_API [lpxError](#) IPXD_CALL [lpxDisplay_Initialize](#) ([lpxHandle](#) hDisplay, void *displayWindow, const char *mode, [lpxImage](#) *imageParams)
This C-interface function initializes the display library for playing videos/still images with the specified mode and image parameters.
- IPXD_EXTERN_C IPXD_API [lpxError](#) IPXD_CALL [lpxDisplay_DisplayVideo](#) ([lpxHandle](#) hDisplay, const [lpxImage](#) *pImage)
This C-interface function displays the video frame.
- IPXD_EXTERN_C IPXD_API [lpxError](#) IPXD_CALL [lpxDisplay_DisplayImage](#) ([lpxHandle](#) hDisplay, const [lpxImage](#) *pImage, const char *mode)
This C-interface function displays the still image.
- IPXD_EXTERN_C IPXD_API [lpxError](#) IPXD_CALL [lpxDisplay_ConvertImage](#) ([lpxHandle](#) hDisplay, const [lpxImage](#) *pSrc, [lpxImage](#) *pDst)
This C-interface function converts the input source [lpxImage](#) to the targeted output destination.

4.31.1 Detailed Description

C-interface functions for [lpxDisplay](#).

4.31.2 Function Documentation

4.31.2.1 IpxDisplay_CreateComponent()

```
IPXD_EXTERN_C IPXD_API IpxHandle IPXD_CALL IpxDisplay_CreateComponent ( )
```

This C-interface function returns the IpxHandle for the created [IpxDisplay](#) instance.

Returns

Returns the IpxHandle for the created [IpxDisplay](#) object

Here is the caller graph for this function:



4.31.2.2 IpxDisplay_DeleteComponent()

```
IPXD_EXTERN_C IPXD_API void IPXD_CALL IpxDisplay_DeleteComponent (
    IpxHandle hDisplay )
```

This C-interface function deletes the IpxHandle `hDisplay` component and all associated resources obtained by the [IpxDisplay](#) object.

Parameters

in	<i>hDisplay</i>	Pointer to the IpxHandle for the IpxDisplay instance
----	-----------------	--

Returns

Void

Here is the caller graph for this function:

**4.31.2.3 IpxDisplay_GetComponent()**

```
IPXD_EXTERN_C IPXD_API IpxComponent* IPXD_CALL IpxDisplay_GetComponent (
    IpxHandle hDisplay )
```

This C-interface function returns the `IpxHandle` for the created `IpxImageConverter` instance.

Parameters

in	<i>hDisplay</i>	Pointer to the <code>IpxHandle</code> for the <code>IpxDisplay</code> instance
----	-----------------	--

Returns

Returns the `IpxHandle` for the `IpxDisplay` object component

Here is the caller graph for this function:



4.31.2.4 IpxDisplay_Initialize()

```
IPXD_EXTERN_C IPXD_API IpxError IPXD_CALL IpxDisplay_Initialize (
    IpxHandle hDisplay,
    void * displayWindow,
    const char * mode,
    IpxImage * imageParams )
```

This C-interface function initializes the display library for playing videos/still images with the specified mode and image parameters.

Parameters

in	<i>hDisplay</i>	pointer to the IpxHandle for the IpxDisplay instance
in	<i>displayWindow</i>	pointer to window. If the displayWindow is not specified, it will create a window.
in	<i>mode</i>	Display mode ("GDI", "OpenGL" mode or "auto" (default) for auto-detection)
in	<i>imageParams</i>	pointer to Image Parameters

Returns

Returns an error code:

- If successful, the IpxError code is IPX_ERR_OK and the display library has been initialized.
- Otherwise, the initialization of the display library failed.

Parameters

in	<i>hDisplay</i>	Pointer to the IpxHandle for the IpxDisplay
----	-----------------	---

Here is the caller graph for this function:



4.31.2.5 IpxDisplay_DisplayVideo()

```
IPXD_EXTERN_C IPXD_API IpxError IPXD_CALL IpxDisplay_DisplayVideo (
    IpxHandle hDisplay,
    const IpxImage * pImage )
```


This C-interface function displays the video frame.

Parameters

in	<i>hDisplay</i>	pointer to the <code>IpxHandle</code> for the <code>IpxDisplay</code> instance
in	<i>pImage</i>	Pointer to the <code>IpxImage</code>

Returns

Returns an error code:

- If successful, the `IpxError` code is `IPX_ERR_OK` and the function displays the video frame.
- Otherwise, the video frame is not displayed

Here is the caller graph for this function:

**4.31.2.6 IpxDisplay_DisplayImage()**

```

IPXD_EXTERN_C IPXD_API IpxError IPXD_CALL IpxDisplay_DisplayImage (
    IpxHandle hDisplay,
    const IpxImage * pImage,
    const char * mode )
  
```

This C-interface function displays the still image.

Parameters

in	<i>hDisplay</i>	pointer to the <code>IpxHandle</code> for the <code>IpxDisplay</code> instance
in	<i>pImage</i>	Pointer to the <code>IpxImage</code> image
in	<i>mode</i>	Display mode ("GDI", "OpenGL" mode or "auto" (default) for auto-detection)

Returns

Returns an error code:

- If successful, the `IpxError` code is `IPX_ERR_OK` and the function displays the still image.
- Otherwise, the video frame is not displayed.

Here is the caller graph for this function:



4.31.2.7 IpxDisplay_ConvertImage()

```

IPXD_EXTERN_C IPXD_API IpxError IPXD_CALL IpxDisplay_ConvertImage (
    IpxHandle hDisplay,
    const IpxImage * pSrc,
    IpxImage * pDst )
  
```

This C-interface function converts the input source `IpxImage` to the targeted output destination.

Parameters

in	<code>hDisplay</code>	Pointer to the <code>IpxHandle</code> for the <code>IpxDisplay</code>
in	<code>pSrc</code>	Pointer to the source <code>IpxImage</code>
out	<code>pDst</code>	Pointer to the output destination <code>IpxImage</code>

Returns

Returns the error code:

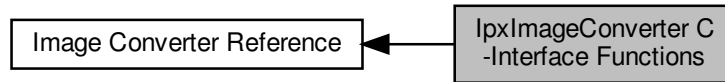
- `IPX_ERR_OK` Successfully converts the source `IpxImage` to the targeted output destination `IpxImage`
- If `IpxError` error code < 0, then it returns a negative error code indicating problems converting the `IpxImage`

Here is the caller graph for this function:



4.32 lpxImageConverter C-Interface Functions

Collaboration diagram for lpxImageConverter C-Interface Functions:



Functions

- IPXC_EXTERN_C IPXC_API [lpxHandle](#) IPXC_CALL [lpxImageConverter_CreateComponent](#) ()
This C-interface function returns the [lpxHandle](#) for the created [lpxImageConverter](#) instance.
- IPXC_EXTERN_C IPXC_API void IPXC_CALL [lpxImageConverter_DeleteComponent](#) ([lpxHandle](#) hImageConverter)
This C-interface function deletes the [lpxHandle](#) hImageConverter component and all associated resources obtained by the [lpxImageConverter](#) object.
- IPXC_EXTERN_C IPXC_API [lpxHandle](#) IPXC_CALL [lpxImageConverter_GetComponent](#) ([lpxHandle](#) hImageConverter)
This C-interface function returns the [lpxHandle](#) for the created [lpxImageConverter](#) instance.
- IPXC_EXTERN_C IPXC_API [lpxError](#) IPXC_CALL [lpxImageConverter_ConvertImage](#) ([lpxHandle](#) hImageConverter, [lpxImage](#) *source, [lpxImage](#) *output)
This C-interface function converts the input source [lpxImage](#) to the targeted output destination.
- IPXC_EXTERN_C IPXC_API [lpxError](#) IPXC_CALL [lpxImageConverter_IICovert](#) ([lpxHandle](#) hImageConverter, [lpxImage](#) *image_in, unsigned long outPixelType, [lpxImage](#) **image_out)
This C-interface function converts the input source [lpxImage](#) to the targeted output destination [lpxImage](#) based on the output pixel type.

4.32.1 Detailed Description

Additional documentation for C-interface functions for '[lpxImageConverter](#)'

4.32.2 Function Documentation

4.32.2.1 IpxImageConverter_CreateComponent()

```
IPXC_EXTERN_C IPXC_API IpxHandle IPXC_CALL IpxImageConverter_CreateComponent ( )
```

This C-interface function returns the IpxHandle for the created [IpxImageConverter](#) instance.

Returns

Returns the IpxHandle for the created [IpxImageConverter](#) object

4.32.2.2 IpxImageConverter_DeleteComponent()

```
IPXC_EXTERN_C IPXC_API void IPXC_CALL IpxImageConverter_DeleteComponent (
    IpxHandle hImageConverter )
```

This C-interface function deletes the IpxHandle hImageConverter component and all associated resources obtained by the [IpxImageConverter](#) object.

Parameters

in	<i>hImageConverter</i>	Pointer to the IpxHandle for the IpxImageConverter instance
----	------------------------	---

Returns

void

4.32.2.3 IpxImageConverter_GetComponent()

```
IPXC_EXTERN_C IPXC_API IpxHandle IPXC_CALL IpxImageConverter_GetComponent (
    IpxHandle hImageConverter )
```

This C-interface function returns the IpxHandle for the created [IpxImageConverter](#) instance.

Parameters

in	<i>hImageConverter</i>	Pointer to the IpxHandle for the IpxImageConverter instance
----	------------------------	---

Returns

Returns the IpxHandle for the [IpxImageConverter](#) object component

4.32.2.4 IpxImageConverter_ConvertImage()

```
IPXC_EXTERN_C IPXC_API IpxError IPXC_CALL IpxImageConverter_ConvertImage (
    IpxHandle hImageConverter,
    IpxImage * source,
    IpxImage * output )
```

This C-interface function converts the input source [IpxImage](#) to the targeted output destination.

Parameters

in	<i>hImageConverter</i>	Pointer to the IpxHandle for the IpxImage Converter
in	<i>source</i>	Pointer to the source IpxImage
out	<i>output</i>	Pointer to the output destination IpxImage

Returns

Returns the error code:

- IPX_ERR_OK Successfully converts the source [IpxImage](#) to the targeted output destination [IpxImage](#)
- If IpxError error code < 0, then it returns a negative error code indicating problems converting the [IpxImage](#)

4.32.2.5 IpxImageConverter_IICovert()

```
IPXC_EXTERN_C IPXC_API IpxError IPXC_CALL IpxImageConverter_IICovert (
    IpxHandle hImageConverter,
    IpxImage * image_in,
    unsigned long outPixelFormat,
    IpxImage ** image_out )
```

This C-interface function converts the input source [IpxImage](#) to the targeted output destination [IpxImage](#) based on the output pixel type.

Parameters

in	<i>hImageConverter</i>	Pointer of the IpxHandle for the IpxImage Converter
in	<i>image_in</i>	Pointer to the source IpxImage
in	<i>outPixelFormat</i>	Output pixel type
out	<i>image_out</i>	Pointer to the output destination IpxImage

Returns

Returns the error code:

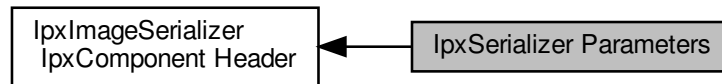
- IPX_ERR_OK Successfully converts the source [IpxImage](#) to the targeted output destination [IpxImage](#) based on the output pixel type.

- If lpxError error code < 0 , then it returns a negative error code indicating problems converting the [lpxImage](#)

4.33 IpxSerializer Parameters

Defines for IpxSerializer Parameters.

Collaboration diagram for IpxSerializer Parameters:



Macros

- #define `ISP_NO_REALLOC` "NoRealloc"
- #define `ISP_JPEG_QUALITY` "jpeg.quality"
- #define `ISP_MIN_QUANTIZER` "min.quantizer"
- #define `ISP_MAX_QUANTIZER` "max.quantizer"
- #define `ISP_TICKS_PER_SEC` "ticks.per.sec"
- #define `ISP_MOVIE_COMPRESSOR` "movie.compressor"
- #define `ISP_MOVIE_COMPRESSORS` "movie.compressors"
- #define `ISP_ADD_PALETTE` "add.palette"

4.33.1 Detailed Description

Defines for IpxSerializer Parameters.

Table 4.65 IpxSerializer Parameters

Macro	Parameter Name	Type	Description
<code>ISP_NO_REALLOC</code>	"NoRealloc"	[int: 0, 1]	does not allow to realloc buffers on runtime
<code>ISP_JPEG_QUALITY</code>	"jpeg.quality"	[int: 1,100]	jpeg quality (1..100; 85 by default)
<code>ISP_MIN_QUANTIZER</code>	"min.quantizer"	[int]	codec minimum quantizer (15 by default, -1 means codec's default value)
<code>ISP_MAX_QUANTIZER</code>	"max.quantizer"	[int]	codec maximum quantizer (15 by default, -1 means codec's default value)
<code>ISP_TICKS_PER_SEC</code>	"ticks.per.sec"	[int]	meaning of timestamp ticks (1000000000 by default, means timestamp in nanoseconds)
<code>ISP_MOVIE_COMPRESSOR</code>	"movie.compressor"	[char*]	movie compressor ("Uncompressed" by default)
<code>ISP_MOVIE_COMPRESSORS</code>	"movie.compressors"	[char*]	list of available compressors separated by
<code>ISP_ADD_PALETTE</code>	"add.palette"	[int: 0, 1]	add palette to the header if image pixeltype is 8 bit grayscale (default 0)

4.33.2 Macro Definition Documentation

4.33.2.1 ISP_NO_REALLOC

```
#define ISP_NO_REALLOC "NoRealloc"
```

Does not allow to realloc buffers on runtime

Type/Range [int: 0, 1]

Note

Used by SetParamInt and GetParamInt

4.33.2.2 ISP_JPEG_QUALITY

```
#define ISP_JPEG_QUALITY "jpeg.quality"
```

Jpeg quality (1..100; 85 by default)

Type/Range [int: 1,100]

Note

Used by SetParamInt and GetParamInt

4.33.2.3 ISP_MIN_QUANTIZER

```
#define ISP_MIN_QUANTIZER "min.quantizer"
```

Codec minimum quantizer (15 by default, -1 means codec's default value will be used)

Type [int]

Note

Used by SetParamInt and GetParamInt

4.33.2.4 ISP_MAX_QUANTIZER

```
#define ISP_MAX_QUANTIZER "max.quantizer"
```

Codec maximum quantizer (15 by default, -1 means codec's default value will be used)

Type [int]

Note

Used by SetParamInt and GetParamInt

4.33.2.5 ISP_TICKS_PER_SEC

```
#define ISP_TICKS_PER_SEC "ticks.per.sec"
```

Meaning of timestamp ticks (1000000000 by default, means timestamp in nanoseconds)

Type [int]

Note

Used by SetParamInt and GetParamInt

4.33.2.6 ISP_MOVIE_COMPRESSOR

```
#define ISP_MOVIE_COMPRESSOR "movie.compressor"
```

Movie compressor (char*; "Uncompressed" by default)

Type/Range [char*]

Note

Used by SetParamString and GetParamString

4.33.2.7 ISP_MOVIE_COMPRESSORS

```
#define ISP_MOVIE_COMPRESSORS "movie.compressors"
```

List of available compressors separated by |

Type/Range [char*]

Note

Used by SetParamString and GetParamString

4.33.2.8 ISP_ADD_PALETTE

```
#define ISP_ADD_PALETTE "add.palette"
```

Add palette to the header if image pixeltype is 8 bit grayscale (default 0)

Type/Range [int: 0, 1]

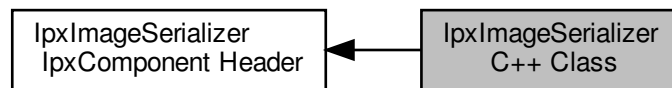
Note

Used by SetParamInt and GetParamInt

4.34 IpXImageSerializer C++ Class

C++ Class for [IpXImageSerializer](#).

Collaboration diagram for IpXImageSerializer C++ Class:



Classes

- class [IpXImageSerializer](#)
IpXComponent to save *IpXImage* to disk.

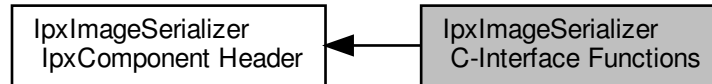
4.34.1 Detailed Description

C++ Class for [IpXImageSerializer](#).

4.35 lpxImageSerializer C-Interface Functions

C-interface functions for [lpxImageSerializer](#).

Collaboration diagram for lpxImageSerializer C-Interface Functions:



Functions

- IPXS_EXTERN_C IPXS_API [lpxHandle](#) IPXS_CALL [lpxImageSerializer_CreateComponent](#) (bool enableMovies)
This C-interface function returns the lpxHandle for the created [lpxImageSerializer](#) instance.
- IPXS_EXTERN_C IPXS_API void IPXS_CALL [lpxImageSerializer_DeleteComponent](#) ([lpxHandle](#) hImage↔
 Serializer)
This C-interface function returns the lpxHandle for the created [lpxImageSerializer](#) instance.
- IPXS_EXTERN_C IPXS_API [lpxHandle](#) IPXS_CALL [lpxImageSerializer_GetComponent](#) ([lpxHandle](#) hImage↔
 Serializer)
This C-interface function returns the lpxHandle for the created [lpxImageSerializer](#) instance.
- IPXS_EXTERN_C IPXS_API [lpxError](#) IPXS_CALL [lpxImageSerializer_StartSeriesRecord](#) ([lpxHandle](#) hImage↔
 Serializer, [lpxImage](#) *pSrc, const char *format)
This C-interface function starts the series record.
- IPXS_EXTERN_C IPXS_API [lpxError](#) IPXS_CALL [lpxImageSerializer_StartMovieRecord](#) ([lpxHandle](#) hImage↔
 Serializer, [lpxImage](#) *pSrc, const char *fileName, double fps)
This C-interface function starts the series record.
- IPXS_EXTERN_C IPXS_API [lpxError](#) IPXS_CALL [lpxImageSerializer_FinishRecord](#) ([lpxHandle](#) hImage↔
 Serializer)
This C-interface function finishes the record.
- IPXS_EXTERN_C IPXS_API [lpxError](#) IPXS_CALL [lpxImageSerializer_Save](#) ([lpxHandle](#) hImageSerializer, [lpx↔
 Image](#) *image, const char *fileName)
This C-interface function saves the record.
- IPXS_EXTERN_C IPXS_API [lpxError](#) IPXS_CALL [lpxImageSerializer_Load](#) ([lpxHandle](#) hImageSerializer, [lpx↔
 Image](#) *image, const char *fileName)
This C-interface function loads the record.

4.35.1 Detailed Description

C-interface functions for [lpxImageSerializer](#).

4.35.2 Function Documentation

4.35.2.1 IpxImageSerializer_CreateComponent()

```
IPXS_EXTERN_C IPXS_API IpxHandle IPXS_CALL IpxImageSerializer_CreateComponent (
    bool enableMovies )
```

This C-interface function returns the IpxHandle for the created [IpxImageSerializer](#) instance.

Parameters

in	<i>enableMovies</i>	flag to enable Movies
----	---------------------	-----------------------

Returns

Returns the IpxHandle for the created [IpxImageSerializer](#) object

4.35.2.2 IpxImageSerializer_DeleteComponent()

```
IPXS_EXTERN_C IPXS_API void IPXS_CALL IpxImageSerializer_DeleteComponent (
    IpxHandle hImageSerializer )
```

This C-interface function returns the IpxHandle for the created [IpxImageSerializer](#) instance.

Parameters

in	<i>hImageSerializer</i>	Pointer to the IpxHandle for the IpxImageSerializer instance
----	-------------------------	--

Returns

void

4.35.2.3 IpxImageSerializer_GetComponent()

```
IPXS_EXTERN_C IPXS_API IpxHandle IPXS_CALL IpxImageSerializer_GetComponent (
    IpxHandle hImageSerializer )
```

This C-interface function returns the IpxHandle for the created [IpxImageSerializer](#) instance.

Parameters

in	<i>hImageSerializer</i>	Pointer to the IpxHandle for the IpxImageSerializer instance
----	-------------------------	--

Returns

Returns the IpxHandle for the [IpxImageSerializer](#) object component

4.35.2.4 IpxImageSerializer_StartSeriesRecord()

```
IPXS_EXTERN_C IPXS_API IpxError IPXS_CALL IpxImageSerializer_StartSeriesRecord (
    IpxHandle hImageSerializer,
    IpxImage * pSrc,
    const char * format )
```

This C-interface function starts the series record.

Parameters

in	<i>hImageSerializer</i>	Pointer to the IpxHandle for the IpxImageSerializer instance
in	<i>pSrc</i>	input source Imperx Image
in	<i>format</i>	Image Format Type

Returns

Returns the error code:

- IPX_ERR_OK Successfully starts the series record.
- If IpxError error code < 0, then it returns a negative error code indicating problem starting the series record

4.35.2.5 IpxImageSerializer_StartMovieRecord()

```
IPXS_EXTERN_C IPXS_API IpxError IPXS_CALL IpxImageSerializer_StartMovieRecord (
    IpxHandle hImageSerializer,
    IpxImage * pSrc,
    const char * fileName,
    double fps )
```

This C-interface function starts the series record.

Parameters

in	<i>hImageSerializer</i>	Pointer to the <i>IpxHandle</i> for the IpxImageSerializer instance
in	<i>pSrc</i>	input source Imperx Image
in	<i>fileName</i>	file name
in	<i>fps</i>	frames per second

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully starts movie record.
- If `IpxError` error code < 0 , then it returns a negative error code indicating problem starting the movie record

4.35.2.6 IpxImageSerializer_FinishRecord()

```
IPXS_EXTERN_C IPXS_API IpxError IPXS_CALL IpxImageSerializer_FinishRecord (
    IpxHandle hImageSerializer )
```

This C-interface function finishes the record.

Parameters

in	<i>hImageSerializer</i>	Pointer to the <i>IpxHandle</i> for the IpxImageSerializer instance
----	-------------------------	---

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully finishes the record.
- If `IpxError` error code < 0 , then it returns a negative error code indicating problem finishing the record

4.35.2.7 IpxImageSerializer_Save()

```
IPXS_EXTERN_C IPXS_API IpxError IPXS_CALL IpxImageSerializer_Save (
    IpxHandle hImageSerializer,
    IpxImage * image,
    const char * fileName )
```

This C-interface function saves the record.

Parameters

in	<i>hImageSerializer</i>	Pointer to the IpxHandle for the IpxImageSerializer instance
in	<i>image</i>	input source Imperx Image
in	<i>fileName</i>	file name

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully saves the record
- If IpxError error code < 0, then it returns a negative error code indicating problem saving the record

4.35.2.8 IpxImageSerializer_Load()

```
IPXS_EXTERN_C IPXS_API IpxError IPXS_CALL IpxImageSerializer_Load (  
    IpxHandle hImageSerializer,  
    IpxImage * image,  
    const char * fileName )
```

This C-interface function loads the record.

Parameters

in	<i>hImageSerializer</i>	Pointer to the IpxHandle for the IpxImageSerializer instance
in	<i>image</i>	input source Imperx Image
in	<i>fileName</i>	file name

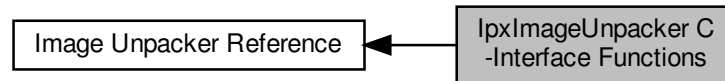
Returns

Returns the error code:

- `IPX_ERR_OK` Successfully allocates the [IpxImage](#) data.
- If IpxError error code < 0, then it returns a negative error code indicating problems allocating [IpxImage](#) data

4.36 lpxImageUnpacker C-Interface Functions

Collaboration diagram for lpxImageUnpacker C-Interface Functions:



Functions

- IPXU_EXTERN_C IPXU_API [lpxHandle](#) IPXU_CALL [lpxImageUnpacker_CreateComponent](#) ()
This C-interface function returns the [lpxHandle](#) for the created [lpxImageUnpacker](#) instance.
- IPXU_EXTERN_C IPXU_API void IPXU_CALL [lpxImageUnpacker_DeleteComponent](#) ([lpxHandle](#) hImageUnpacker)
This C-interface function deletes the [lpxHandle](#) hImageUnpacker component and all associated resources obtained by the [lpxImageUnpacker](#) object.
- IPXU_EXTERN_C IPXU_API [lpxHandle](#) IPXU_CALL [lpxImageUnpacker_GetComponent](#) ([lpxHandle](#) hImageUnpacker)
This C-interface function returns the [lpxHandle](#) for the created [lpxImageUnpacker](#) instance.
- IPXU_EXTERN_C IPXU_API [lpxError](#) IPXU_CALL [lpxImageUnpacker_Unpack](#) ([lpxHandle](#) hImageUnpacker, [lpxImage](#) *source, [lpxImage](#) *output)
This C-interface function unpacks the input raw source [lpxRAWImage](#) to the targeted output destination [lpxImage](#).

4.36.1 Detailed Description

Additional documentation for C-interface functions for '[lpxImageUnpacker](#)'

4.36.2 Function Documentation

4.36.2.1 lpxImageUnpacker_CreateComponent()

```
IPXU_EXTERN_C IPXU_API lpxHandle IPXU_CALL lpxImageUnpacker_CreateComponent ( )
```

This C-interface function returns the [lpxHandle](#) for the created [lpxImageUnpacker](#) instance.

Returns

Returns the [lpxHandle](#) for the created [lpxImageUnpacker](#) object

4.36.2.2 IpxImageUnpacker_DeleteComponent()

```
IPXU_EXTERN_C IPXU_API void IPXU_CALL IpxImageUnpacker_DeleteComponent (
    IpxHandle hImageUnpacker )
```

This C-interface function deletes the IpxHandle hImageUnpacker component and all associated resources obtained by the [IpxImageUnpacker](#) object.

Parameters

in	<i>hImageUnpacker</i>	Pointer to the IpxHandle for the IpxImageUnpacker instance
----	-----------------------	--

Returns

void

4.36.2.3 IpxImageUnpacker_GetComponent()

```
IPXU_EXTERN_C IPXU_API IpxHandle IPXU_CALL IpxImageUnpacker_GetComponent (
    IpxHandle hImageUnpacker )
```

This C-interface function returns the IpxHandle for the created [IpxImageUnpacker](#) instance.

Parameters

in	<i>hImageUnpacker</i>	Pointer to the IpxHandle for the IpxImageUnpacker instance
----	-----------------------	--

Returns

Returns the IpxHandle for the [IpxImageUnpacker](#) object component

4.36.2.4 IpxImageUnpacker_Unpack()

```
IPXU_EXTERN_C IPXU_API IpxError IPXU_CALL IpxImageUnpacker_Unpack (
    IpxHandle hImageUnpacker,
    IpxImage * source,
    IpxImage * output )
```

This C-interface function unpacks the input raw source IpxRAWImage to the targeted output destination [IpxImage](#).

Parameters

in	<i>hImageUnpacker</i>	Pointer of the lpxHandle for the lpxUnpacker
in	<i>source</i>	Pointer to the raw source lpxRawImage
out	<i>output</i>	Pointer to the output destination lpxImage

Returns

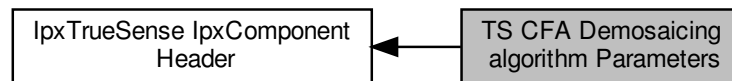
Returns the error code:

- `IPX_ERR_OK` Successfully unpacks the source [lpxImage](#) to the targeted output destination [lpxImage](#).
- If lpxError error code < 0 , then it returns a negative error code indicating problems unpacking the [lpxImage](#)

4.37 TS CFA Demosaicing algorithm Parameters

Defines for TS CFA Demosaicing algorithms.

Collaboration diagram for TS CFA Demosaicing algorithm Parameters:



Macros

- `#define TS_ALGO_TYPE "TrueSenseAlgType"`
- `#define TS_NOREALLOC "NoRealloc"`
- `#define TS_ALGO_NUM 7`
- `#define TSASIMPLEF 0`
- `#define TSASIMPLES 1`
- `#define TSABAYERLIKE 2`
- `#define TSAMEDIMUM 3`
- `#define TSAQUALITY 4`
- `#define TRUES_OPENGL_MHC 5`
- `#define TRUES_OPENGL_MMA 6`

4.37.1 Detailed Description

Defines for TS CFA Demosaicing algorithms.

Table 4.77 TS CFA Demosaicing Algorithm Parameters

Macro	Parameter Name	Type	Description
TS_ALGO_TYPE	"TrueSenseAlgType"	[int: 0, TS_ALGO_NUM - 1]	TrueSense Algorithm Type
TS_NOREALLOC	"NoRealloc"	[int: 0,1]	NoRealloc Enabled

4.37.2 Macro Definition Documentation

4.37.2.1 TS_ALGO_TYPE

```
#define TS_ALGO_TYPE "TrueSenseAlgType"
```

TrueSense Algorithm Type

Type/Range [int: 0, TS_ALGO_NUM - 1]

Note

Used by SetParamInt and GetParamInt

4.37.2.2 TS_NOREALLOC

```
#define TS_NOREALLOC "NoRealloc"
```

NoRealloc Enabled

Type/Range [int: 0,1]

Note

Used by SetParamInt and GetParamInt

4.37.2.3 TS_ALGO_NUM

```
#define TS_ALGO_NUM 7
```

Number of Algorithms Supported

4.37.2.4 TSASIMPLEF

```
#define TSASIMPLEF 0
```

Simple algorithm. Average quality, high speed. You can set tonescale table in TrueSenseParam structure.

4.37.2.5 TSASIMPLES

```
#define TSASIMPLES 1
```

Simple Quality algorithm. Average quality, high speed. You can set tonescale table and white balance coefficients in TrueSenseParam structure.

4.37.2.6 TSABAYERLIKE

```
#define TSABAYERLIKE 2
```

Simple Bayer-like algorithm. Average quality, high speed. You can set tonescale table in TrueSenseParam structure.

4.37.2.7 TSAMEDIUM

```
#define TSAMEDIUM 3
```

High Quality algorithm. High quality, medium speed. You can set all of the adjusting parameters in TrueSenseParam structure.

4.37.2.8 TSAQUALITY

```
#define TSAQUALITY 4
```

High Quality algorithm. High quality, very low speed. You can set all of the adjusting parameters in TrueSenseParam structure.

4.37.2.9 TRUES_OPENGL_MHC

```
#define TRUES_OPENGL_MHC 5
```

OpenGL MHC algorithm.

4.37.2.10 TRUES_OPENGL_MMA

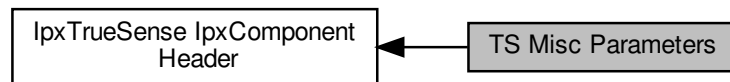
```
#define TRUES_OPENGL_MMA 6
```

OpenGL MMA algorithm.

4.38 TS Misc Parameters

Defines for TS Misc parameters.

Collaboration diagram for TS Misc Parameters:



Macros

- `#define TS_THREADS_NUM "threads_num"`
- `#define TS_NORM_EN "normalizationEnable"`
- `#define TS_HORIZ_MIRRORED "horMirrored"`
- `#define TS_VER_MIRRORED "verMirrored"`
- `#define TS_MONO_ENABLED "monoEnable"`
- `#define TS_IMP_FILTER_ENABLED "impulseFilterEnable"`
- `#define TS_SHARPNESS_ENABLED "sharpnessEnable"`
- `#define TS_DARKFLOOR "darkFloor"`

4.38.1 Detailed Description

Defines for TS Misc parameters.

Table 4.78 TS Misc Parameters

Macro	Parameter Name	Type	Description
TS_THREADS_NUM	"threads_num"	[int: 0-32]	Quantity of threads used in calculation. Default value is 0, it means maximum number of available threads
TS_NORM_EN	"normalizationEnable"	[int: 0,1]	Enable normalization. 0 - off, 1 - on. Default value is off
TS_HORIZ_MIRRORED	"horMirrored"	[int: 0,1]	If image is mirrored horizontally. Default value is 0.
TS_VER_MIRRORED	"verMirrored"	[int: 0,1]	If image is mirrored vertically. Default value is 0.
TS_MONO_ENABLED	"monoEnable"	[int: 0,1]	Switch on monochrome processing instead of color processing. 0 - color, 1 - monochrome. Reserved.
TS_IMP_FILTER_ENABLED	"impulseFilterEnable"	[int: 0,1]	Enable the impulse filter processing. 0 - off, 1 - on.
TS_SHARPNESS_ENABLED	"sharpnessEnable"	[int: 0,1]	Enable the sharpness processing. 0 - off, 1 - on.
TS_DARKFLOOR	"darkFloor"	[int: 0-4096]	Dark floor of raw image, fetched from raw file header.

4.38.2 Macro Definition Documentation

4.38.2.1 TS_THREADS_NUM

```
#define TS_THREADS_NUM "threads_num"
```

Quantity of threads used in calculation. Default value is 0, it means maximum number of available threads

Type/Range [int: 0-32]

Note

Used by SetParamInt and GetParamInt

4.38.2.2 TS_NORM_EN

```
#define TS_NORM_EN "normalizationEnable"
```

Enable normalization. 0 - off, 1 - on. Default value is off.

Type/Range [int: 0,1]

Note

Used by SetParamInt and GetParamInt

4.38.2.3 TS_HORIZ_MIRRORED

```
#define TS_HORIZ_MIRRORED "horMirrored"
```

If image is mirrored horizontally. Default value is 0.

Type/Range [int: 0,1]

Note

Used by SetParamInt and GetParamInt

4.38.2.4 TS_VER_MIRRORED

```
#define TS_VER_MIRRORED "verMirrored"
```

If image is mirrored vertically. Default value is 0.

Type/Range [int: 0,1]

Note

Used by SetParamInt and GetParamInt

4.38.2.5 TS_MONO_ENABLED

```
#define TS_MONO_ENABLED "monoEnable"
```

Switch on monochrome processing instead of color processing. 0 - color, 1 - monochrome. Reserved.

Type/Range [int: 0,1]

Note

Used by SetParamInt and GetParamInt

4.38.2.6 TS_IMP_FILTER_ENABLED

```
#define TS_IMP_FILTER_ENABLED "impulseFilterEnable"
```

Enable the impulse filter processing. 0 - off, 1 - on.

Type/Range [int: 0,1]

Note

Used by SetParamInt and GetParamInt

4.38.2.7 TS_SHARPNESS_ENABLED

```
#define TS_SHARPNESS_ENABLED "sharpnessEnable"
```

Enable the sharpness processing. 0 - off, 1 - on.

Type/Range [int: 0,1]

Note

Used by SetParamInt and GetParamInt

4.38.2.8 TS_DARKFLOOR

```
#define TS_DARKFLOOR "darkFloor"
```

Dark floor of raw image, fetched from raw file header.

Type/Range [int: 0,1]

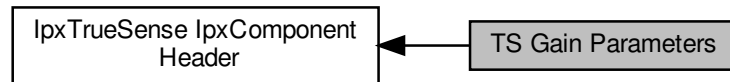
Note

Used by SetParamInt and GetParamInt

4.39 TS Gain Parameters

Defines for TS gain parameters.

Collaboration diagram for TS Gain Parameters:



Macros

- `#define TS_RED_GAIN "redGain"`
- `#define TS_GREEN_GAIN "greenGain"`
- `#define TS_BLUE_GAIN "blueGain"`
- `#define TS_PAN_GAIN "panGain"`
- `#define TS_GLOBAL_GAIN "globalGain"`
- `#define TS_ANALOG_GAIN "analogGain"`
- `#define TS_ISO_ANALOGGAIN_0 "ISOAnalogGain_0"`
- `#define TS_ISO_ANALOGGAIN_1 "ISOAnalogGain_1"`
- `#define TS_ISO_ANALOGGAIN_2 "ISOAnalogGain_2"`
- `#define TS_ISO_ANALOGGAIN_3 "ISOAnalogGain_3"`
- `#define TS_ISO_ANALOGGAIN_4 "ISOAnalogGain_4"`

4.39.1 Detailed Description

Defines for TS gain parameters.

Table 4.79 TS Gain Parameters

Macro	Parameter Name	Type	Description
TS_RED_GAIN	"redGain"	[float: DBL_MIN-DBL_MAX]	Red gain of white balance.
TS_GREEN_GAIN	"greenGain"	[float: DBL_MIN-DBL_MAX]	Green gain of white balance.
TS_BLUE_GAIN	"blueGain"	[float: DBL_MIN-DBL_MAX]	Blue gain of white balance.
TS_PAN_GAIN	"panGain"	[float: DBL_MIN-DBL_MAX]	Panchromatic gain of white balance. It should be set as 1 currently.
TS_GLOBAL_GAIN	"globalGain"	[float: DBL_MIN-DBL_MAX]	Digital gain. It will be applied to processing if more than 1.0

Macro	Parameter Name	Type	Description
TS_ANALOG_GAIN	"analogGain"	[float: DBL_MIN-DBL_MAX]	Actual sensor gain of raw image, fetched from raw file header.
TS_ISO_ANALOGGAIN_0	"ISOAnalogGain↔ _0"	[float: DBL_MIN-DBL_MAX]	Sensor gain array of typical ISO levels, used to interpolate intermediate noise variation slope and intercept.
TS_ISO_ANALOGGAIN_1	"ISOAnalogGain↔ _1"	[float: DBL_MIN-DBL_MAX]	Sensor gain array of typical ISO levels, used to interpolate intermediate noise variation slope and intercept.
TS_ISO_ANALOGGAIN_2	"ISOAnalogGain↔ _2"	[float: DBL_MIN-DBL_MAX]	Sensor gain array of typical ISO levels, used to interpolate intermediate noise variation slope and intercept.
TS_ISO_ANALOGGAIN_3	"ISOAnalogGain↔ _3"	[float: DBL_MIN-DBL_MAX]	Sensor gain array of typical ISO levels, used to interpolate intermediate noise variation slope and intercept.
TS_ISO_ANALOGGAIN_4	"ISOAnalogGain↔ _4"	[float: DBL_MIN-DBL_MAX]	Sensor gain array of typical ISO levels, used to interpolate intermediate noise variation slope and intercept.

4.39.2 Macro Definition Documentation

4.39.2.1 TS_RED_GAIN

```
#define TS_RED_GAIN "redGain"
```

Red gain of white balance.

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat

4.39.2.2 TS_GREEN_GAIN

```
#define TS_GREEN_GAIN "greenGain"
```

Green gain of white balance.

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat

4.39.2.3 TS_BLUE_GAIN

```
#define TS_BLUE_GAIN "blueGain"
```

Blue gain of white balance.

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat

4.39.2.4 TS_PAN_GAIN

```
#define TS_PAN_GAIN "panGain"
```

Panchromatic gain of white balance. It should be set as 1 currently.

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat

4.39.2.5 TS_GLOBAL_GAIN

```
#define TS_GLOBAL_GAIN "globalGain"
```

Digital gain. It will be applied to processing if more than 1.0.

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat

4.39.2.6 TS_ANALOG_GAIN

```
#define TS_ANALOG_GAIN "analogGain"
```

Actual sensor gain of raw image, fetched from raw file header.

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat

4.39.2.7 TS_ISO_ANALOGGAIN_0

```
#define TS_ISO_ANALOGGAIN_0 "ISOAnalogGain_0"
```

Sensor gain array of typical ISO levels

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, used to interpolate intermediate noise variation slope and intercept.
Ex: ISO400_AnalogGain = 11.04 dB

4.39.2.8 TS_ISO_ANALOGGAIN_1

```
#define TS_ISO_ANALOGGAIN_1 "ISOAnalogGain_1"
```

Sensor gain array of typical ISO levels

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, used to interpolate intermediate noise variation slope and intercept.
Ex: ISO800_AnalogGain = 17.69 dB

4.39.2.9 TS_ISO_ANALOGGAIN_2

```
#define TS_ISO_ANALOGGAIN_2 "ISOAnalogGain_2"
```

Sensor gain array of typical ISO levels

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, used to interpolate intermediate noise variation slope and intercept.
Ex: ISO1600_AnalogGain = 23.96 dB

4.39.2.10 TS_ISO_ANALOGGAIN_3

```
#define TS_ISO_ANALOGGAIN_3 "ISOAnalogGain_3"
```

Sensor gain array of typical ISO levels

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, used to interpolate intermediate noise variation slope and intercept.
Ex: ISO3200_AnalogGain = 29.91 dB

4.39.2.11 TS_ISO_ANALOGGAIN_4

```
#define TS_ISO_ANALOGGAIN_4 "ISOAnalogGain_4"
```

Sensor gain array of typical ISO levels

Type/Range [float: DBL_MIN-DBL_MAX]

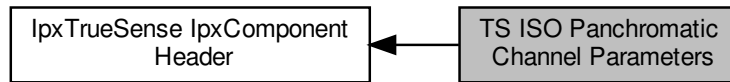
Note

Used by SetParamFloat and GetParamFloat, used to interpolate intermediate noise variation slope and intercept.
Ex: ISO6400_AnalogGain = 36.77 dB

4.40 TS ISO Panchromatic Channel Parameters

Defines for TS ISO Panchromatic channel parameters.

Collaboration diagram for TS ISO Panchromatic Channel Parameters:



Macros

- `#define TS_ISO_PANSLOPE_0 "ISOPanSlope_0"`
- `#define TS_ISO_PANSLOPE_1 "ISOPanSlope_1"`
- `#define TS_ISO_PANSLOPE_2 "ISOPanSlope_2"`
- `#define TS_ISO_PANSLOPE_3 "ISOPanSlope_3"`
- `#define TS_ISO_PANSLOPE_4 "ISOPanSlope_4"`
- `#define TS_ISO_PANINTERCEPT_0 "ISOPanIntercept_0"`
- `#define TS_ISO_PANINTERCEPT_1 "ISOPanIntercept_1"`
- `#define TS_ISO_PANINTERCEPT_2 "ISOPanIntercept_2"`
- `#define TS_ISO_PANINTERCEPT_3 "ISOPanIntercept_3"`
- `#define TS_ISO_PANINTERCEPT_4 "ISOPanIntercept_4"`

4.40.1 Detailed Description

Defines for TS ISO Panchromatic channel parameters.

Table 4.80 TS ISO Panchromatic Channel Parameters

Macro	Parameter Name	Type	Description
TS_ISO_PANSLOPE_0	"ISOPanSlope_0"	[float: DBL_MIN-DBL_MAX]	Noise variation slope of panchromatic channel at typical sensor gains
TS_ISO_PANSLOPE_1	"ISOPanSlope_1"	[float: DBL_MIN-DBL_MAX]	Noise variation slope of panchromatic channel at typical sensor gains
TS_ISO_PANSLOPE_2	"ISOPanSlope_2"	[float: DBL_MIN-DBL_MAX]	Noise variation slope of panchromatic channel at typical sensor gains
TS_ISO_PANSLOPE_3	"ISOPanSlope_3"	[float: DBL_MIN-DBL_MAX]	Noise variation slope of panchromatic channel at typical sensor gains

Macro	Parameter Name	Type	Description
TS_ISO_PANSLOPE_4	"ISOPanSlope_4"	[float: DBL_MIN-DBL_MAX]	Noise variation slope of panchromatic channel at typical sensor gains
TS_ISO_PANINTERCEPT_0 ↔	"ISOPanSlope_0"	[float: DBL_MIN-DBL_MAX]	Noise variation intercept of panchromatic channel at typical sensor gains
TS_ISO_PANINTERCEPT_1 ↔	"ISOPanSlope_1"	[float: DBL_MIN-DBL_MAX]	Noise variation intercept of panchromatic channel at typical sensor gains
TS_ISO_PANINTERCEPT_2 ↔	"ISOPanSlope_2"	[float: DBL_MIN-DBL_MAX]	Noise variation intercept of panchromatic channel at typical sensor gains
TS_ISO_PANINTERCEPT_3 ↔	"ISOPanSlope_3"	[float: DBL_MIN-DBL_MAX]	Noise variation intercept of panchromatic channel at typical sensor gains
TS_ISO_PANINTERCEPT_4 ↔	"ISOPanSlope_4"	[float: DBL_MIN-DBL_MAX]	Noise variation intercept of panchromatic channel at typical sensor gains

4.40.2 Macro Definition Documentation

4.40.2.1 TS_ISO_PANSLOPE_0

```
#define TS_ISO_PANSLOPE_0 "ISOPanSlope_0"
```

Noise variation slope of panchromatic channel at typical sensor gains

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, used to interpolate intermediate noise variation slope. Ex: ISO400↔
_PanSlope = 0.31793097

4.40.2.2 TS_ISO_PANSLOPE_1

```
#define TS_ISO_PANSLOPE_1 "ISOPanSlope_1"
```

Noise variation slope of panchromatic channel at typical sensor gains

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, used to interpolate intermediate noise variation slope. Ex: ISO800↔
_PanSlope = 0.6009852

4.40.2.3 TS_ISO_PANSLOPE_2

```
#define TS_ISO_PANSLOPE_2 "ISOPanSlope_2"
```

Noise variation slope of panchromatic channel at typical sensor gains

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, used to interpolate intermediate noise variation slope. Ex: ISO1600_PanSlope = 1.16587611

4.40.2.4 TS_ISO_PANSLOPE_3

```
#define TS_ISO_PANSLOPE_3 "ISOPanSlope_3"
```

Noise variation slope of panchromatic channel at typical sensor gains

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, used to interpolate intermediate noise variation slope. Ex: ISO3200_PanSlope = 2.26059552

4.40.2.5 TS_ISO_PANSLOPE_4

```
#define TS_ISO_PANSLOPE_4 "ISOPanSlope_4"
```

Noise variation slope of panchromatic channel at typical sensor gains

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, used to interpolate intermediate noise variation slope. Ex: ISO6400_PanSlope = 5.11044291

4.40.2.6 TS_ISO_PANINTERCEPT_0

```
#define TS_ISO_PANINTERCEPT_0 "ISOPanIntercept_0"
```

Noise variation intercept of panchromatic channel at typical sensor gains

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, used to interpolate intermediate noise variation intercept. Ex: ISO400_PanIntercept = -25.07685652

4.40.2.7 TS_ISO_PANINTERCEPT_1

```
#define TS_ISO_PANINTERCEPT_1 "ISOPanIntercept_1"
```

Noise variation intercept of panchromatic channel at typical sensor gains

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, used to interpolate intermediate noise variation intercept. Ex: ISO800_PanIntercept = 17.01752105

4.40.2.8 TS_ISO_PANINTERCEPT_2

```
#define TS_ISO_PANINTERCEPT_2 "ISOPanIntercept_2"
```

Noise variation intercept of panchromatic channel at typical sensor gains

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, used to interpolate intermediate noise variation intercept. Ex: ISO1600_PanIntercept = 185.43026

4.40.2.9 TS_ISO_PANINTERCEPT_3

```
#define TS_ISO_PANINTERCEPT_3 "ISOPanIntercept_3"
```

Noise variation intercept of panchromatic channel at typical sensor gains

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, used to interpolate intermediate noise variation intercept. Ex: IS↔
O3200_PanIntercept = 831.07495077

4.40.2.10 TS_ISO_PANINTERCEPT_4

```
#define TS_ISO_PANINTERCEPT_4 "ISOPanIntercept_4"
```

Noise variation intercept of panchromatic channel at typical sensor gains

Type/Range [float: DBL_MIN-DBL_MAX]

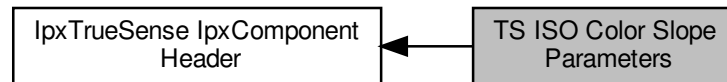
Note

Used by SetParamFloat and GetParamFloat, used to interpolate intermediate noise variation intercept. Ex: IS↔
O6400_PanIntercept = 4154.73883603

4.41 TS ISO Color Slope Parameters

Defines for TS ISO Color Slope parameters.

Collaboration diagram for TS ISO Color Slope Parameters:



Macros

- `#define TS_ISO_COLORSLOPE_0 "ISOCColorSlope_0"`
- `#define TS_ISO_COLORSLOPE_1 "ISOCColorSlope_1"`
- `#define TS_ISO_COLORSLOPE_2 "ISOCColorSlope_2"`
- `#define TS_ISO_COLORSLOPE_3 "ISOCColorSlope_3"`
- `#define TS_ISO_COLORSLOPE_4 "ISOCColorSlope_4"`

4.41.1 Detailed Description

Defines for TS ISO Color Slope parameters.

Table 4.81 TS ISO Color Slope Parameters

Macro	Parameter Name	Type	Description
TS_ISO_COLORSLOPE_0	"ISOCColorSlope↔ _0"	[float: DBL_MIN-DBL_MAX]	Noise variation slope of color channel at typical sensor gains
TS_ISO_COLORSLOPE_1	"ISOCColorSlope↔ _1"	[float: DBL_MIN-DBL_MAX]	Noise variation slope of color channel at typical sensor gains
TS_ISO_COLORSLOPE_2	"ISOCColorSlope↔ _2"	[float: DBL_MIN-DBL_MAX]	Noise variation slope of color channel at typical sensor gains
TS_ISO_COLORSLOPE_3	"ISOCColorSlope↔ _3"	[float: DBL_MIN-DBL_MAX]	Noise variation slope of color channel at typical sensor gains
TS_ISO_COLORSLOPE_4	"ISOCColorSlope↔ _4"	[float: DBL_MIN-DBL_MAX]	Noise variation slope of color channel at typical sensor gains

4.41.2 Macro Definition Documentation

4.41.2.1 TS_ISO_COLORSLOPE_0

```
#define TS_ISO_COLORSLOPE_0 "ISOCOLORSLOPE_0"
```

Noise variation slope of color channel at typical sensor gains

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, used to interpolate intermediate noise variation slope. Ex: ISO400↵
_ColorSlope = 0.16289523

4.41.2.2 TS_ISO_COLORSLOPE_1

```
#define TS_ISO_COLORSLOPE_1 "ISOCOLORSLOPE_1"
```

Noise variation slope of color channel at typical sensor gains

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, used to interpolate intermediate noise variation slope. Ex: ISO800↵
_ColorSlope = 0.30242107

4.41.2.3 TS_ISO_COLORSLOPE_2

```
#define TS_ISO_COLORSLOPE_2 "ISOCOLORSLOPE_2"
```

Noise variation slope of color channel at typical sensor gains

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, used to interpolate intermediate noise variation slope. Ex: IS↵
O1600_ColorSlope = 0.58180185

4.41.2.4 TS_ISO_COLORSLOPE_3

```
#define TS_ISO_COLORSLOPE_3 "ISOCOLORSLOPE_3"
```

Noise variation slope of color channel at typical sensor gains

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, used to interpolate intermediate noise variation slope. Ex: IS↔O3200_ColorSlope = 1.15281985

4.41.2.5 TS_ISO_COLORSLOPE_4

```
#define TS_ISO_COLORSLOPE_4 "ISOCOLORSLOPE_4"
```

Noise variation slope of color channel at typical sensor gains

Type/Range [float: DBL_MIN-DBL_MAX]

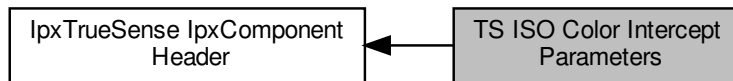
Note

Used by SetParamFloat and GetParamFloat, used to interpolate intermediate noise variation slope. Ex: IS↔O6400_ColorSlope = 2.53400236

4.42 TS ISO Color Intercept Parameters

Defines for TS ISO Color Intercept parameters.

Collaboration diagram for TS ISO Color Intercept Parameters:



Macros

- #define `TS_ISO_COLORINTERCEPT_0` "ISOCOLORINTERCEPT_0"
- #define `TS_ISO_COLORINTERCEPT_1` "ISOCOLORINTERCEPT_1"
- #define `TS_ISO_COLORINTERCEPT_2` "ISOCOLORINTERCEPT_2"
- #define `TS_ISO_COLORINTERCEPT_3` "ISOCOLORINTERCEPT_3"
- #define `TS_ISO_COLORINTERCEPT_4` "ISOCOLORINTERCEPT_4"

4.42.1 Detailed Description

Defines for TS ISO Color Intercept parameters.

Table 4.82 TS ISO Color Intercept Parameters

Macro	Parameter Name	Type	Description
<code>TS_ISO_COLORINTERCEPT_0</code>	"ISOCOLORINTERCEPT_0"	[float: DBL_MIN-DBL_MAX]	Noise variation intercept of color channel at typical sensor gains
<code>TS_ISO_COLORINTERCEPT_1</code>	"ISOCOLORINTERCEPT_1"	[float: DBL_MIN-DBL_MAX]	Noise variation intercept of color channel at typical sensor gains
<code>TS_ISO_COLORINTERCEPT_2</code>	"ISOCOLORINTERCEPT_2"	[float: DBL_MIN-DBL_MAX]	Noise variation intercept of color channel at typical sensor gains
<code>TS_ISO_COLORINTERCEPT_3</code>	"ISOCOLORINTERCEPT_3"	[float: DBL_MIN-DBL_MAX]	Noise variation intercept of color channel at typical sensor gains
<code>TS_ISO_COLORINTERCEPT_4</code>	"ISOCOLORINTERCEPT_4"	[float: DBL_MIN-DBL_MAX]	Noise variation intercept of color channel at typical sensor gains

4.42.2 Macro Definition Documentation

4.42.2.1 TS_ISO_COLORINTERCEPT_0

```
#define TS_ISO_COLORINTERCEPT_0 "ISOCOLORIntercept_0"
```

Noise variation intercept of color channel at typical sensor gains

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, used to interpolate intermediate noise variation intercept. Ex: IS↵O400_ColorIntercept = -2.97408598

4.42.2.2 TS_ISO_COLORINTERCEPT_1

```
#define TS_ISO_COLORINTERCEPT_1 "ISOCOLORIntercept_1"
```

Noise variation intercept of color channel at typical sensor gains

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, used to interpolate intermediate noise variation intercept. Ex: IS↵O800_ColorIntercept = 15.97559859

4.42.2.3 TS_ISO_COLORINTERCEPT_2

```
#define TS_ISO_COLORINTERCEPT_2 "ISOCOLORIntercept_2"
```

Noise variation intercept of color channel at typical sensor gains

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, used to interpolate intermediate noise variation intercept. Ex: IS↵O1600_ColorIntercept = 92.84640595

4.42.2.4 TS_ISO_COLORINTERCEPT_3

```
#define TS_ISO_COLORINTERCEPT_3 "ISOColorIntercept_3"
```

Noise variation intercept of color channel at typical sensor gains

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, used to interpolate intermediate noise variation intercept. Ex: IS↔
O3200_ColorIntercept = 399.49923562

4.42.2.5 TS_ISO_COLORINTERCEPT_4

```
#define TS_ISO_COLORINTERCEPT_4 "ISOColorIntercept_4"
```

Noise variation intercept of color channel at typical sensor gains

Type/Range [float: DBL_MIN-DBL_MAX]

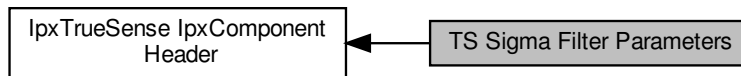
Note

Used by SetParamFloat and GetParamFloat, used to interpolate intermediate noise variation intercept. Ex: IS↔
O6400_ColorIntercept = 2080.24259272

4.43 TS Sigma Filter Parameters

Defines for TS Sigma Filter parameters.

Collaboration diagram for TS Sigma Filter Parameters:



Macros

- `#define TS_PAN_RADIUS0` "panRadius0"
- `#define TS_PAN_RADIUS1` "panRadius1"
- `#define TS_PAN_RADIUS2` "panRadius2"
- `#define TS_PAN_SIGMA0` "panSigma0"
- `#define TS_PAN_SIGMA1` "panSigma1"
- `#define TS_PAN_SIGMA2` "panSigma2"
- `#define TS_COLOR_RADIUS0` "colorRadius0"
- `#define TS_COLOR_RADIUS1` "colorRadius1"
- `#define TS_COLOR_RADIUS2` "colorRadius2"
- `#define TS_COLOR_SIGMA0` "colorSigma0"
- `#define TS_COLOR_SIGMA1` "colorSigma1"
- `#define TS_COLOR_SIGMA2` "colorSigma2"

4.43.1 Detailed Description

Defines for TS Sigma Filter parameters.

Table 4.83 TS Sigma Filter Parameters

Macro	Parameter Name	Type	Description
TS_PAN_RADIUS0	"panRadius0"	[float: DBL_MIN-DBL_MAX]	Pixel radius for the sigma filter of first round panchromatic channel noise cleaning
TS_PAN_RADIUS1	"panRadius1"	[float: DBL_MIN-DBL_MAX]	Pixel radius for the sigma filter of second round panchromatic channel noise cleaning
TS_PAN_RADIUS2	"panRadius2"	[float: DBL_MIN-DBL_MAX]	Pixel radius for the sigma filter of third round panchromatic channel noise cleaning

Macro	Parameter Name	Type	Description
TS_PAN_SIGMA0	"panSigma0"	[float: DBL_MIN-DBL_MAX]	Scalar for the sigma filter of first round panchromatic channel noise cleaning
TS_PAN_SIGMA1	"panSigma1"	[float: DBL_MIN-DBL_MAX]	Scalar for the sigma filter of second round panchromatic channel noise cleaning
TS_PAN_SIGMA2	"panSigma2"	[float: DBL_MIN-DBL_MAX]	Scalar for the sigma filter of third round panchromatic channel noise cleaning
TS_COLOR_RADIUS0	"colorRadius0"	[float: DBL_MIN-DBL_MAX]	Pixel radius for the sigma filter of first round color channel noise cleaning
TS_COLOR_RADIUS1	"colorRadius1"	[float: DBL_MIN-DBL_MAX]	Pixel radius for the sigma filter of second round color channel noise cleaning
TS_COLOR_RADIUS2	"colorRadius2"	[float: DBL_MIN-DBL_MAX]	Pixel radius for the sigma filter of third round color channel noise cleaning
TS_COLOR_SIGMA0	"colorSigma0"	[float: DBL_MIN-DBL_MAX]	Scalar for the sigma filter of first round color channel noise cleaning
TS_COLOR_SIGMA1	"colorSigma1"	[float: DBL_MIN-DBL_MAX]	Scalar for the sigma filter of second round color channel noise cleaning
TS_COLOR_SIGMA2	"colorSigma2"	[float: DBL_MIN-DBL_MAX]	Scalar for the sigma filter of third round color channel noise cleaning

4.43.2 Macro Definition Documentation

4.43.2.1 TS_PAN_RADIUS0

```
#define TS_PAN_RADIUS0 "panRadius0"
```

Pixel radius for the sigma filter of first round panchromatic channel noise cleaning, 0 means bypass current round cleaning.

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat

4.43.2.2 TS_PAN_RADIUS1

```
#define TS_PAN_RADIUS1 "panRadius1"
```

Pixel radius for the sigma filter of second round panchromatic channel noise cleaning, 0 means bypass current round cleaning.

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat

4.43.2.3 TS_PAN_RADIUS2

```
#define TS_PAN_RADIUS2 "panRadius2"
```

Pixel radius for the sigma filter of third round panchromatic channel noise cleaning, 0 means bypass current round cleaning.

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat

4.43.2.4 TS_PAN_SIGMA0

```
#define TS_PAN_SIGMA0 "panSigma0"
```

Scalar for the sigma filter of first round panchromatic channel noise cleaning

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat

4.43.2.5 TS_PAN_SIGMA1

```
#define TS_PAN_SIGMA1 "panSigma1"
```

Scalar for the sigma filter of second round panchromatic channel noise cleaning

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat

4.43.2.6 TS_PAN_SIGMA2

```
#define TS_PAN_SIGMA2 "panSigma2"
```

Scalar for the sigma filter of third round panchromatic channel noise cleaning

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat

4.43.2.7 TS_COLOR_RADIUS0

```
#define TS_COLOR_RADIUS0 "colorRadius0"
```

Pixel radius for the sigma filter of first round color channel noise cleaning, 0 means bypass current round cleaning.

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat

4.43.2.8 TS_COLOR_RADIUS1

```
#define TS_COLOR_RADIUS1 "colorRadius1"
```

Pixel radius for the sigma filter of second round color channel noise cleaning, 0 means bypass current round cleaning.

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat

4.43.2.9 TS_COLOR_RADIUS2

```
#define TS_COLOR_RADIUS2 "colorRadius2"
```

Pixel radius for the sigma filter of third round color channel noise cleaning, 0 means bypass current round cleaning.

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat

4.43.2.10 TS_COLOR_SIGMA0

```
#define TS_COLOR_SIGMA0 "colorSigma0"
```

Scalar for the sigma filter of first round color channel noise cleaning

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat

4.43.2.11 TS_COLOR_SIGMA1

```
#define TS_COLOR_SIGMA1 "colorSigma1"
```

Scalar for the sigma filter of second round color channel noise cleaning

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat

4.43.2.12 TS_COLOR_SIGMA2

```
#define TS_COLOR_SIGMA2 "colorSigma2"
```

Scalar for the sigma filter of third round color channel noise cleaning

Type/Range [float: DBL_MIN-DBL_MAX]

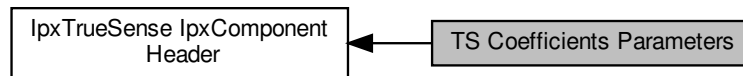
Note

Used by SetParamFloat and GetParamFloat

4.44 TS Coefficients Parameters

Defines for TS Coefficients parameters.

Collaboration diagram for TS Coefficients Parameters:



Macros

- #define **TS_RR_COEFF** "RR"
- #define **TS_RG_COEFF** "RG"
- #define **TS_RB_COEFF** "RB"
- #define **TS_GR_COEFF** "GR"
- #define **TS_GG_COEFF** "GG"
- #define **TS_GB_COEFF** "GB"
- #define **TS_BR_COEFF** "BR"
- #define **TS_BG_COEFF** "BG"
- #define **TS_BB_COEFF** "BB"

4.44.1 Detailed Description

Defines for TS Coefficients parameters.

Table 4.84 TS Coefficients Parameters

Macro	Parameter Name	Type	Description
TS_RR_COEFF	"RR"	[float: DBL_MIN-DBL_MAX]	Red-red coefficient of color correction matrix
TS_RG_COEFF	"RG"	[float: DBL_MIN-DBL_MAX]	Red-green coefficient of color correction matrix
TS_RB_COEFF	"RB"	[float: DBL_MIN-DBL_MAX]	Red-blue coefficient of color correction matrix
TS_GR_COEFF	"GR"	[float: DBL_MIN-DBL_MAX]	Green-red coefficient of color correction matrix
TS_GG_COEFF	"GG"	[float: DBL_MIN-DBL_MAX]	Green-green coefficient of color correction matrix
TS_GB_COEFF	"GB"	[float: DBL_MIN-DBL_MAX]	Green-blue coefficient of color correction matrix
TS_BR_COEFF	"BR"	[float: DBL_MIN-DBL_MAX]	Blue-red coefficient of color correction matrix
TS_BG_COEFF	"BG"	[float: DBL_MIN-DBL_MAX]	Blue-green coefficient of color correction matrix
TS_BB_COEFF	"BB"	[float: DBL_MIN-DBL_MAX]	blue-blue coefficient of color correction matrix

4.44.2 Macro Definition Documentation

4.44.2.1 TS_RR_COEFF

```
#define TS_RR_COEFF "RR"
```

Red-red coefficient of color correction matrix

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, Example: 1.657

4.44.2.2 TS_RG_COEFF

```
#define TS_RG_COEFF "RG"
```

Red-green coefficient of color correction matrix

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, Example: -0.5325

4.44.2.3 TS_RB_COEFF

```
#define TS_RB_COEFF "RB"
```

Red-blue coefficient of color correction matrix

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, Example: -0.1245

4.44.2.4 TS_GR_COEFF

```
#define TS_GR_COEFF "GR"
```

Green-red coefficient of color correction matrix

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, Example: -0.106

4.44.2.5 TS_GG_COEFF

```
#define TS_GG_COEFF "GG"
```

Green-green coefficient of color correction matrix

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, Example: 1.443

4.44.2.6 TS_GB_COEFF

```
#define TS_GB_COEFF "GB"
```

Green-blue coefficient of color correction matrix

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, Example: -0.337

4.44.2.7 TS_BR_COEFF

```
#define TS_BR_COEFF "BR"
```

Blue-red coefficient of color correction matrix

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, Example: 0.131

4.44.2.8 TS_BG_COEFF

```
#define TS_BG_COEFF "BG"
```

Blue-green coefficient of color correction matrix

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat, Example: -0.445

4.44.2.9 TS_BB_COEFF

```
#define TS_BB_COEFF "BB"
```

Blue-blue coefficient of color correction matrix

Type/Range [float: DBL_MIN-DBL_MAX]

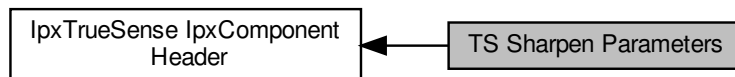
Note

Used by SetParamFloat and GetParamFloat, Example: 1.314

4.45 TS Sharpen Parameters

Defines for TS Sharpen parameters.

Collaboration diagram for TS Sharpen Parameters:



Macros

- `#define TS_SHARPEN_PARAM "sharpenParam"`
- `#define TS_MAX_SHARPEN "maxSharpen"`

4.45.1 Detailed Description

Defines for TS Sharpen parameters.

Table 4.85 TS Sharpen Parameters

Macro	Parameter Name	Type	Description
TS_SHARPEN_PARAM	"sharpenParam"	[float: DBL_MIN-DBL_MAX]	Sharp parameter
TS_MAX_SHARPEN	"maxSharpen"	[float: DBL_MIN-DBL_MAX]	Sharp maximal threshold

4.45.2 Macro Definition Documentation

4.45.2.1 TS_SHARPEN_PARAM

```
#define TS_SHARPEN_PARAM "sharpenParam"
```

Sharp parameter

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat

4.45.2.2 TS_MAX_SHARPEN

```
#define TS_MAX_SHARPEN "maxSharpen"
```

Sharp maximal threshold

Type/Range [float: DBL_MIN-DBL_MAX]

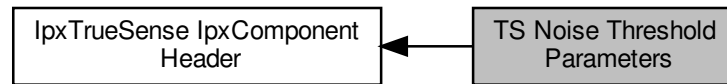
Note

Used by SetParamFloat and GetParamFloat

4.46 TS Noise Threshold Parameters

Defines for TS Noise Threshold parameters.

Collaboration diagram for TS Noise Threshold Parameters:



Macros

- `#define TS_HIGH_LUMA_NOISE "highLumaNoise"`
- `#define TS_LOW_LUMA_NOISE "lowLumaNoise"`

4.46.1 Detailed Description

Defines for TS Noise Threshold parameters.

Table 4.86 TS Noise Threshold Parameters

Macro	Parameter Name	Type	Description
TS_SHARPEN_PARAM	"highLumaNoise"	[float: DBL_MIN-DBL_MAX]	High Noise threshold
TS_MAX_SHARPEN	"lowLumaNoise"	[float: DBL_MIN-DBL_MAX]	Low Noise threshold

4.46.2 Macro Definition Documentation

4.46.2.1 TS_HIGH_LUMA_NOISE

```
#define TS_HIGH_LUMA_NOISE "highLumaNoise"
```

High Noise threshold

Type/Range [float: DBL_MIN-DBL_MAX]

Note

Used by SetParamFloat and GetParamFloat

4.46.2.2 TS_LOW_LUMA_NOISE

```
#define TS_LOW_LUMA_NOISE "lowLumaNoise"
```

Low Noise threshold

Type/Range [float: DBL_MIN-DBL_MAX]

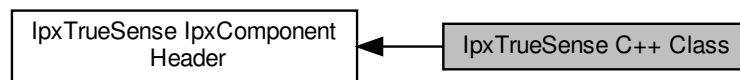
Note

Used by SetParamFloat and GetParamFloat

4.47 IpxTrueSense C++ Class

C++ Class for [IpxTrueSense](#).

Collaboration diagram for IpxTrueSense C++ Class:



Classes

- class [IpxTrueSense](#)

A Class for [IpxTrueSense](#) modules that contains methods to convert [IpxImage](#) images.

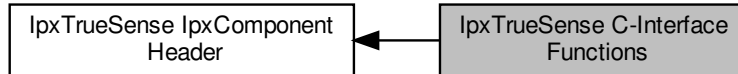
4.47.1 Detailed Description

C++ Class for [IpxTrueSense](#).

4.48 lpxTrueSense C-Interface Functions

C-interface functions for [lpxTrueSense](#).

Collaboration diagram for lpxTrueSense C-Interface Functions:



Functions

- TS_EXTERN_C TS_API [lpxHandle](#) TS_CALL [lpxTrueSense_CreateComponent](#) ()
This C-interface function returns the lpxHandle for the created lpxTrueSense instance.
- TS_EXTERN_C TS_API void TS_CALL [lpxTrueSense_DeleteComponent](#) ([lpxHandle](#) hTrueSense)
This C-interface function deletes the lpxHandle hTrueSense component and all associated resources obtained by the [lpxTrueSense](#) object.
- TS_EXTERN_C TS_API [lpxHandle](#) TS_CALL [lpxTrueSense_GetComponent](#) ([lpxHandle](#) hTrueSense)
This C-interface function returns the lpxHandle for the [lpxTrueSense](#) component.
- TS_EXTERN_C TS_API [lpxError](#) TS_CALL [lpxTrueSense_ConvertImage](#) ([lpxHandle](#) hTrueSense, const [lpxImage](#) *pSrc, [lpxImage](#) *pDst)
This C-interface function converts the input source [lpxImage](#) to the targeted output destination.
- TS_EXTERN_C TS_API [lpxError](#) TS_CALL [lpxTrueSense_AllocData](#) ([lpxHandle](#) hTrueSense, const [lpxImage](#) *pSrc, [lpxImage](#) *pDst)
This C-interface function allocates the data.
- TS_EXTERN_C TS_API void TS_CALL [lpxTrueSense_ReleaseData](#) ([lpxHandle](#) hTrueSense)
This C-interface function release the lpxHandle to the [lpxTrueSense](#) data.

4.48.1 Detailed Description

C-interface functions for [lpxTrueSense](#).

4.48.2 Function Documentation

4.48.2.1 IpxTrueSense_CreateComponent()

```
TS_EXTERN_C TS_API IpxHandle TS_CALL IpxTrueSense_CreateComponent ( )
```

This C-interface function returns the IpxHandle for the created IpxTrueSense instance.

Returns

Returns the IpxHandle for the created [IpxTrueSense](#) object
This C-interface function returns the IpxHandle for the created [IpxTrueSense](#) instance

Returns the IpxHandle for the created [IpxTrueSense](#) object

4.48.2.2 IpxTrueSense_DeleteComponent()

```
TS_EXTERN_C TS_API void TS_CALL IpxTrueSense_DeleteComponent (
    IpxHandle hTrueSense )
```

This C-interface function deletes the IpxHandle hTrueSense component and all associated resources obtained by the [IpxTrueSense](#) object.

Parameters

in	<i>hTrueSense</i>	Pointer to the IpxHandle for the IpxTrueSense instance
----	-------------------	--

Returns

void

4.48.2.3 IpxTrueSense_GetComponent()

```
TS_EXTERN_C TS_API IpxHandle TS_CALL IpxTrueSense_GetComponent (
    IpxHandle hTrueSense )
```

This C-interface function returns the IpxHandle for the [IpxTrueSense](#) component.

Parameters

in	<i>hTrueSense</i>	Pointer to the IpxHandle for the IpxTrueSense object
----	-------------------	--

Returns

Returns the IpxHandle for the [IpxTrueSense](#) component

4.48.2.4 IpxTrueSense_ConvertImage()

```
TS_EXTERN_C TS_API IpxError TS_CALL IpxTrueSense_ConvertImage (
    IpxHandle hTrueSense,
    const IpxImage * pSrc,
    IpxImage * pDst )
```

This C-interface function converts the input source [IpxImage](#) to the targeted output destination.

Parameters

in	<i>hTrueSense</i>	Pointer to the IpxHandle for the IpxTrueSense Component
in	<i>pSrc</i>	Pointer to the source IpxImage
out	<i>pDst</i>	Pointer to the output destination IpxImage

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully converts the source [IpxImage](#) to the targeted output destination [IpxImage](#)
- If IpxError error code < 0, then it returns a negative error code indicating problems converting the [IpxImage](#)

4.48.2.5 IpxTrueSense_AllocData()

```
TS_EXTERN_C TS_API IpxError TS_CALL IpxTrueSense_AllocData (
    IpxHandle hTrueSense,
    const IpxImage * pSrc,
    IpxImage * pDst )
```

This C-interface function allocates the data.

Parameters

in	<i>hTrueSense</i>	Pointer of the IpxHandle for the IpxTrueSense Component
in	<i>pSrc</i>	Pointer to the source IpxImage
in	<i>pDst</i>	Pointer to the output destination IpxImage

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully allocates the `lpxImage` data.
- If `lpxError` error code < 0 , then it returns a negative error code indicating problems allocating `lpxImage` data

4.48.2.6 `lpxTrueSense_ReleaseData()`

```
TS_EXTERN_C TS_API void TS_CALL IpxTrueSense_ReleaseData (  
    IpxHandle hTrueSense )
```

This C-interface function release the `lpxHandle` to the `lpxTrueSense` data.

Parameters

in	<i>hTrueSense</i>	Pointer of the <code>lpxHandle</code> for the <code>lpxTrueSense</code> data
----	-------------------	--

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully releases the `lpxTrueSense` data.
- If `lpxError` error code < 0 , then it returns a negative error code indicating problems releasing the `lpxTrueSense` data

Chapter 5

Class Documentation

5.1 lpxBayer Class Reference

A Class for [lpxBayer](#) modules that contains methods to convert Bayer CFA (Color Filter Array) images.

```
#include <IpxBayer.h>
```

Public Member Functions

- virtual [lpxComponent](#) * [GetComponent](#) ()=0
This function returns the pointer to the [lpxComponent](#) object.
- virtual [lpxError](#) [ConvertImage](#) (const [lpxImage](#) *pSrc, [lpxImage](#) *pDst)=0
This function Bayer CFA (Color Filter Array) Demosaicing converts the input source [lpxImage](#) to the targeted output destination [lpxImage](#).
- virtual [lpxError](#) [AllocData](#) (const [lpxImage](#) *pSrc, [lpxImage](#) *pDst)=0
This function allocates memory.
- virtual void [ReleaseData](#) ()=0
This function releases the allocated memory.

Static Public Member Functions

- static BAYER_API [lpxBayer](#) * [CreateComponent](#) ()
This function returns the created [lpxBayer](#) instance.
- static BAYER_API void [DeleteComponent](#) ([lpxBayer](#) *in)
This function deletes the [lpxBayer](#) component and all associated resources obtained by the [lpxBayer](#) object.

5.1.1 Detailed Description

A Class for [lpxBayer](#) modules that contains methods to convert Bayer CFA (Color Filter Array) images.

A class containing methods for [lpxBayer](#) modules.

5.1.2 Member Function Documentation

5.1.2.1 CreateComponent()

```
static BAYER_API IpxBayer* IpxBayer::CreateComponent ( ) [static]
```

This function returns the created [IpxBayer](#) instance.

Returns

Returns the created [IpxBayer](#) object

5.1.2.2 DeleteComponent()

```
static BAYER_API void IpxBayer::DeleteComponent (
    IpxBayer * in ) [static]
```

This function deletes the [IpxBayer](#) component and all associated resources obtained by the [IpxBayer](#) object.

Parameters

in	in	Pointer to the IpxBayer object
----	----	--

Returns

Returns void

5.1.2.3 GetComponent()

```
virtual IpxComponent* IpxBayer::GetComponent ( ) [pure virtual]
```

This function returns the pointer to the [IpxComponent](#) object.

The [IpxComponent](#) object will give access to the data member functions shown below:

- 🔒 GetComponentTypeID
- 🔒 GetParamAsString
- 🔒 GetParamBool
- 🔒 GetParamCount
- 🔒 GetParamFloat
- 🔒 GetParamInt
- 🔒 GetParamName
- 🔒 GetParamString
- 🔒 RunCommand
- 🔒 SetParamAsString
- 🔒 SetParamBool
- 🔒 SetParamFloat
- 🔒 SetParamInt
- 🔒 SetParamString
- 🔒 ~IpxComponent

Returns

Returns the Pointer to the [IpxComponent](#) object

The following example will illustrate on how to access the [IpxComponent](#) data member function:

```
//Create the IpxBayer Component
IpxBayer *pDeBayer = IpxBayer::CreateComponent();

//Access the set parameter data member function using the GetComponent function.
//Set the "BayerAlgType" parameter to '2'.
pDeBayer->GetComponent()->SetParamInt("BayerAlgType", 2);
```

5.1.2.4 ConvertImage()

```
virtual IpxError IpxBayer::ConvertImage (
    const IpxImage * pSrc,
    IpxImage * pDst ) [pure virtual]
```

This function Bayer CFA (Color Filter Array) Demosaicing converts the input source [IpxImage](#) to the targeted output destination [IpxImage](#).

Parameters

in	pSrc	Pointer to the input source IpxlImage
----	------	---

The only input source Pixel Types supported are shown below:

Table 5.3 Input Source Supported Types

Pixel Types
II_PIX_BAYGR8
II_PIX_BAYGR10
II_PIX_BAYGR12
II_PIX_BAYGR14
II_PIX_BAYGR16
II_PIX_BAYRG8
II_PIX_BAYRG10
II_PIX_BAYRG12
II_PIX_BAYRG14
II_PIX_BAYRG16
II_PIX_BAYBG8
II_PIX_BAYBG10
II_PIX_BAYBG12
II_PIX_BAYBG14
II_PIX_BAYBG16
II_PIX_BAYGB8
II_PIX_BAYGB10
II_PIX_BAYGB12
II_PIX_BAYGB14
II_PIX_BAYGB16

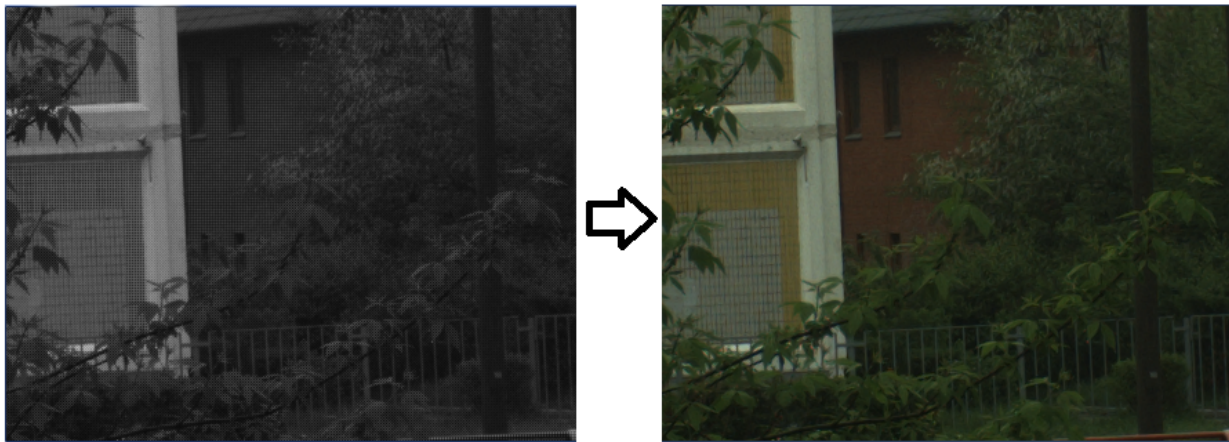


Figure 5.1 Example of a Bayer Conversion Process

Parameters

out	<i>pDst</i>	Pointer to the output destination IpxImage
-----	-------------	--

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully converts the source [IpxImage](#) to the targeted output destination [IpxImage](#)
- If `IpxError` error code < 0 , then it returns a negative error code indicating problems converting the [IpxImage](#)

5.1.2.5 AllocData()

```
virtual IpxError IpxBayer::AllocData (
    const IpxImage * pSrc,
    IpxImage * pDst ) [pure virtual]
```

This function allocates memory.

Parameters

in	<i>pSrc</i>	Pointer to the input source IpxImage
in	<i>pDst</i>	Pointer to the output destination IpxImage

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully allocates data
- If `IpxError` error code < 0 , then it returns a negative error code indicating problem allocating memory

5.1.2.6 ReleaseData()

```
virtual void IpxBayer::ReleaseData ( ) [pure virtual]
```

This function releases the allocated memory.

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully releases the allocated data
- If `IpxError` error code < 0 , then it returns a negative error code indicating problem releasing the data allocated

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

- `IpxBayer.h`

5.2 IpxComponent Class Reference

A Class for [IpxComponent](#) modules that contains methods for setting/getting/executing Component features.

```
#include <IpxToolsBase.h>
```

Public Member Functions

- virtual [~IpxComponent](#) ()
This function releases the resources obtained by the [IpxComponent](#) object.
- virtual [uint8_t GetComponentTypeID](#) ()=0
This function returns the component type ID.
- virtual [size_t GetParamCount](#) ()=0
This function returns the parameter count of the [IpxComponent](#).
- virtual [IpxError GetParamName](#) (uint32_t index, char *name, uint32_t *size)=0
This function returns the parameter name associated with the index.
- virtual [IpxError GetParamAsString](#) (const char *name, char *param, uint32_t *size, const char *format=nullptr)=0
This function gets the requested data information for the corresponding parameter name. This output information is a 'char' type variable.
- virtual [IpxError SetParamAsString](#) (const char *name, char *param)=0
This function sets the named parameter with the parameter data information. The parameter data information is a 'char' type variable.
- virtual [IpxError SetParamBool](#) (const char *name, bool param)=0
This function sets the named bool parameter with the bool parameter data information. The parameter data information is a 'boolean' type variable.
- virtual [IpxError SetParamInt](#) (const char *name, int64_t param)=0
This function sets the named integer parameter with the parameter data information. The parameter data information is a 'int64_t' type variable.
- virtual [IpxError SetParamFloat](#) (const char *name, double param)=0
This function sets the named float parameter with the parameter data information. The parameter data information is a 'double' type variable.
- virtual [IpxError SetParamString](#) (const char *name, char *param)=0
This function sets the named string parameter with the parameter data information. The parameter data information is in char string format.
- virtual [IpxError SetParamArray](#) (const char *name, void *param, uint32_t size)=0
This function sets the named array parameter with the parameter data information. The parameter data information is pointer to memory buffer.
- virtual [IpxError GetParamBool](#) (const char *name, bool *param)=0
This function retrieves the bool parameter data information for the specified named parameter.
- virtual [IpxError GetParamInt](#) (const char *name, int64_t *param)=0
This function retrieves the integer parameter data information for the specified named parameter.
- virtual [IpxError GetParamFloat](#) (const char *name, double *param)=0
This function retrieves the float parameter data information for the specified named parameter.
- virtual [IpxError GetParamString](#) (const char *name, char *param, uint32_t *size)=0
This function retrieves the string parameter data information for the specified named parameter.
- virtual [IpxError GetParamArray](#) (const char *name, void *param, uint32_t *size)=0
This function retrieves the string parameter data information for the specified named parameter.
- virtual [IpxError RunCommand](#) (const char *name)=0
This function runs the command parameter specified.

5.2.1 Detailed Description

A Class for [IpxComponent](#) modules that contains methods for setting/getting/executing Component features.

A class containing methods for [IpxComponent](#) modules.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 ~IpxComponent()

```
virtual IpxComponent::~~IpxComponent ( ) [inline], [virtual]
```

This function releases the resources obtained by the [IpxComponent](#) object.

Returns

Destructor of [IpxComponent](#)

5.2.3 Member Function Documentation

5.2.3.1 GetComponentTypeID()

```
virtual uint8_t IpxComponent::GetComponentTypeID ( ) [pure virtual]
```

This function returns the component type ID.

Returns

This function returns the component type ID.

5.2.3.2 GetParamCount()

```
virtual size_t IpxComponent::GetParamCount ( ) [pure virtual]
```

This function returns the parameter count of the [IpxComponent](#).

Returns

Returns the parameter count

5.2.3.3 GetParamName()

```
virtual IpxError IpxComponent::GetParamName (
    uint32_t index,
    char * name,
    uint32_t * size ) [pure virtual]
```

This function returns the parameter name associated with the index.

Parameters

in	<i>index</i>	Parameter index
out	<i>name</i>	Name of parameter
in	<i>size</i>	input size

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully gets the name of the parameter of the specified index
- If `IpxError` error code < 0 , then it returns a negative error code indicating the named parameter was not found

5.2.3.4 GetParamAsString()

```
virtual IpxError IpxComponent::GetParamAsString (
    const char * name,
    char * param,
    uint32_t * size,
    const char * format = nullptr ) [pure virtual]
```

This function gets the requested data information for the corresponding parameter name. This output information is a 'char' type variable.

Parameters

in	<i>name</i>	Parameter name
out	<i>param</i>	Name of parameter value that was requested and returned in a char string format.
in	<i>size</i>	Input size
in	<i>format</i>	Format of Int to String conversion (default is "%i" PRIi64)

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully gets the name parameter requested data information of the specified index
- If `IpxError` code < 0 , then it returns a negative error code indicating the named parameter was not found

5.2.3.5 SetParamAsString()

```
virtual IpxError IpxComponent::SetParamAsString (
    const char * name,
    char * param ) [pure virtual]
```

This function sets the named parameter with the parameter data information. The parameter data information is a 'char' type variable.

Parameters

in	<i>name</i>	Parameter name
in	<i>param</i>	Parameter value that is being set. The parameter data information is in a char string format.

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully sets the name parameter requested data information specified
- If `IpxError` code < 0 , then it returns a negative error code indicating the named parameter was not found

5.2.3.6 SetParamBool()

```
virtual IpxError IpxComponent::SetParamBool (
    const char * name,
    bool param ) [pure virtual]
```

This function sets the named bool parameter with the bool parameter data information. The parameter data information is a 'boolean' type variable.

Parameters

in	<i>name</i>	Parameter name
in	<i>param</i>	Parameter value that is being set. The parameter data information is a 'boolean' type variable.

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully sets the name parameter requested data information specified
- If `IpxError` code < 0 , then it returns a negative error code indicating the named parameter was not found

5.2.3.7 SetParamInt()

```
virtual IpxError IpxComponent::SetParamInt (
    const char * name,
    int64_t param ) [pure virtual]
```

This function sets the named integer parameter with the parameter data information. The parameter data information is a 'int64_t' type variable.

Parameters

in	<i>name</i>	Parameter name
in	<i>param</i>	Parameter value that is being set. The parameter data information is a 'int64_t' type variable.

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully sets the name parameter requested data information specified
- If `IpxError` code < 0 , then it returns a negative error code indicating the named parameter was not found

5.2.3.8 SetParamFloat()

```
virtual IpxError IpxComponent::SetParamFloat (
    const char * name,
    double param ) [pure virtual]
```

This function sets the named float parameter with the parameter data information. The parameter data information is a 'double' type variable.

Parameters

in	<i>name</i>	Parameter name
in	<i>param</i>	Parameter value that is being set. The parameter data information is a 'double' type variable.

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully sets the name parameter requested data information specified
- If `IpxError` code < 0 , then it returns a negative error code indicating the named parameter was not found

5.2.3.9 SetParamString()

```
virtual IpxError IpxComponent::SetParamString (
    const char * name,
    char * param ) [pure virtual]
```

This function sets the named string parameter with the parameter data information. The parameter data information is in char string format.

Parameters

in	<i>name</i>	Parameter name
in	<i>param</i>	Parameter value that is being set. The parameter data information is in a char string format.

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully sets the name parameter requested data information specified
- If `IpxError` code < 0 , then it returns a negative error code indicating the named parameter was not found

5.2.3.10 SetParamArray()

```
virtual IpxError IpxComponent::SetParamArray (  
    const char * name,  
    void * param,  
    uint32_t size ) [pure virtual]
```

This function sets the named array parameter with the parameter data information. The parameter data information is pointer to memory buffer.

Parameters

in	<i>name</i>	Parameter name
in	<i>param</i>	Parameter value that is being set. The parameter is pointer to memory buffer.
in	<i>size</i>	Size of the memory buffer, specified in param argument, in bytes.

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully sets the name parameter requested data information specified
- If `IpxError` code < 0 , then it returns a negative error code indicating the named parameter was not found

5.2.3.11 GetParamBool()

```
virtual IpxError IpxComponent::GetParamBool (  
    const char * name,  
    bool * param ) [pure virtual]
```

This function retrieves the bool parameter data information for the specified named parameter.

Parameters

in	<i>name</i>	Name of parameter
out	<i>param</i>	retrieves the bool parameter data information

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully gets the named bool parameter data information
- If `IpxError` error code < 0 , then it returns a negative error code indicating the named parameter was not found

5.2.3.12 GetParamInt()

```
virtual IpxError IpxComponent::GetParamInt (
    const char * name,
    int64_t * param ) [pure virtual]
```

This function retrieves the integer parameter data information for the specified named parameter.

Parameters

in	<i>name</i>	Name of parameter
out	<i>param</i>	retrieves the integer parameter data information

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully gets the named integer parameter data information
- If `IpxError` error code < 0 , then it returns a negative error code indicating the named parameter was not found

5.2.3.13 GetParamFloat()

```
virtual IpxError IpxComponent::GetParamFloat (
    const char * name,
    double * param ) [pure virtual]
```

This function retrieves the float parameter data information for the specified named parameter.

Parameters

in	<i>name</i>	Name of parameter
out	<i>param</i>	retrieves the float parameter data information

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully gets the named float parameter data information
- If `IpxError` error code < 0 , then it returns a negative error code indicating the named parameter was not found

5.2.3.14 GetParamString()

```
virtual IpxError IpxComponent::GetParamString (
    const char * name,
    char * param,
    uint32_t * size ) [pure virtual]
```

This function retrieves the string parameter data information for the specified named parameter.

Parameters

in	<i>name</i>	Name of parameter
out	<i>param</i>	retrieves the string parameter data information
in	<i>size</i>	Size of param string being retrieved.

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully gets the named string parameter data information
- If `IpxError` error code < 0 , then it returns a negative error code indicating the named parameter was not found

5.2.3.15 GetParamArray()

```
virtual IpxError IpxComponent::GetParamArray (
    const char * name,
    void * param,
    uint32_t * size ) [pure virtual]
```

This function retrieves the string parameter data information for the specified named parameter.

Parameters

in	<i>name</i>	Name of parameter
out	<i>param</i>	Pointer to memory buffer for parameter data information
in	<i>size</i>	Size of the memory buffer being retrieved, in bytes

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully gets the named string parameter data information
- If `lpxError` error code < 0 , then it returns a negative error code indicating the named parameter was not found

5.2.3.16 RunCommand()

```
virtual IpXError IpXComponent::RunCommand (  
    const char * name ) [pure virtual]
```

This function runs the command parameter specified.

Parameters

in	<i>name</i>	Name of parameter
----	-------------	-------------------

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully runs the command parameter specified
- If `lpxError` error code < 0 , then it returns a negative error code indicating the named parameter was not found

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

- `lpxToolsBase.h`

5.3 IpXDisplay Class Reference

A Class for [IpXDisplay](#) modules that contains methods to display [IpXImage](#) images. This class is responsible for displaying video frames and still images.

```
#include <IpXDisplay.h>
```

Public Member Functions

- virtual [lpxComponent](#) * [GetComponent](#) ()=0
This function returns the pointer to the [lpxComponent](#) object. [QT: yes].
- virtual bool [GetSystemInfo](#) (char *buffer, int32_t bufferSz, const char *separator="; ")=0
This function returns GPU information as text.
- virtual [lpxError](#) [Initialize](#) (void *displayWindow, const char *mode="auto", [lpxImage](#) *imageParams=0)=0
This function initializes the display library for playing videos/still images with the specified mode and image parameters.
- virtual [lpxError](#) [SetVideoMode](#) ([lpxImage](#) *imageParams, const char *mode="auto")=0
- virtual [lpxError](#) [DisplayVideo](#) ([lpxImage](#) *image)=0
This function displays the video frame. [QT: yes].
- virtual [lpxError](#) [DisplayImage](#) ([lpxImage](#) *image, const char *mode="auto")=0
This function displays the still image. [QT: no].
- virtual [lpxError](#) [ConvertImage](#) ([lpxImage](#) *source, [lpxImage](#) *output)=0
This function converts the source image to the specified output image. [QT: no].
- virtual [lpxError](#) [Translate](#) (int32_t *x, int32_t *y, int32_t flags)
This function translates the display object to the specified coordinates as indicated by the flag. [QT: yes].

Static Public Member Functions

- static IPXD_API [lpxDisplay](#) * [CreateComponent](#) ()
This function creates a [lpxComponent](#) and returns the created [lpxDisplay](#) instance [QT: yes].
- static IPXD_API void [DeleteComponent](#) ([lpxDisplay](#) *component)
This function deletes the [lpxDisplay](#) component and all associated resources obtained by the [lpxDisplay](#) object. [QT: yes].

5.3.1 Detailed Description

A Class for [lpxDisplay](#) modules that contains methods to display [lpxImage](#) images. This class is responsible for displaying video frames and still images.

A class containing methods for [lpxDisplay](#) modules.

5.3.2 Member Function Documentation

5.3.2.1 CreateComponent()

```
static IPXD_API lpxDisplay* lpxDisplay::CreateComponent ( ) [static]
```

This function creates a [lpxComponent](#) and returns the created [lpxDisplay](#) instance [QT: yes].

Returns

Returns the created [lpxDisplay](#) object

5.3.2.2 DeleteComponent()

```
static IPXD_API void IpxDisplay::DeleteComponent (
    IpxDisplay * component ) [static]
```

This function deletes the [IpxDisplay](#) component and all associated resources obtained by the [IpxDisplay](#) object. [QT: yes].

Parameters

in	<i>component</i>	Pointer to the IpxDisplay object
----	------------------	--

Returns
















Returns void

5.3.2.3 GetComponent()

```
virtual IpxComponent* IpxDisplay::GetComponent ( ) [pure virtual]
```

This function returns the pointer to the [IpxComponent](#) object. [QT: yes].

The [IpxComponent](#) object will give access to the data member functions shown below:

-  GetComponentTypeID
-  GetParamAsString
-  GetParamBool
-  GetParamCount
-  GetParamFloat
-  GetParamInt
-  GetParamName
-  GetParamString
-  RunCommand
-  SetParamAsString
-  SetParamBool
-  SetParamFloat
-  SetParamInt
-  SetParamString
-  ~IpxComponent

Returns

Returns the Pointer to the [IpxComponent](#) object

The following example will illustrate on how to access the [IpxComponent](#) data member function:

```
IpxDisplay* m_IpxDisplay = IpxDisplay::CreateComponent();
...
//Sets the IDP_OGL_BAYER
m_IpxDisplay->GetComponent()->SetParamString(IDP_OGL_BAYER, "0");
...
```

Note

Please reference the [Display Component Parameters](#) section to view the supported parameter names for [IpxDisplay](#)

5.3.2.4 GetSystemInfo()

```
virtual bool IpxDisplay::GetSystemInfo (
    char * buffer,
    int32_t bufferSz,
    const char * separator = "; " ) [pure virtual]
```

This function returns GPU information as text.

Parameters

in	<i>buffer</i>	allocated buffer for information
in	<i>bufferSz</i>	size of the buffer
in	<i>separator</i>	optional parameters separator

Returns

- If successful, returns true
- Otherwise, returns false

5.3.2.5 Initialize()

```
virtual IpxError IpxDisplay::Initialize (
    void * displayWindow,
```

```
const char * mode = "auto",  
IpxImage * imageParams = 0 ) [pure virtual]
```

This function initializes the display library for playing videos/still images with the specified mode and image parameters.

Parameters

in	<i>displayWindow</i>	pointer to window. If the displayWindow is not specified, it will create a window.
in	<i>mode</i>	Display mode ("GDI", "OpenGL" mode or "auto" (default) for auto-detection)
in	<i>imageParams</i>	pointer to Image Parameters

Returns

Returns an error code:

- If successful, the IpxError code is IPX_ERR_OK and the display library has been initialized.
- Otherwise, the initialization of the display library failed.

5.3.2.6 SetVideoMode()

```
virtual IpxError IpxDisplay::SetVideoMode (  
    IpxImage * imageParams,  
    const char * mode = "auto" ) [pure virtual]
```

This function initializes video player for specified image parameters It should be called each time image parameters has been changed

Parameters

in	<i>imageParams</i>	image with specified width, height and pixel type
in	<i>mode</i>	either "GDI", "OpenGL" mode or "auto" (default) for auto-detection

Returns

Returns an error code:

- If successful, the IpxError code is IPX_ERR_OK and the display is ready to display new pixel format.
- Otherwise, the call has been failed and successive [DisplayVideo\(\)](#) calls will not display any video.

5.3.2.7 DisplayVideo()

```
virtual IpxError IpxDisplay::DisplayVideo (  
    IpxImage * image ) [pure virtual]
```

This function displays the video frame. [QT: yes].

Parameters

in	<i>image</i>	source image
----	--------------	--------------

Returns

Returns an error code:

- If successful, the `IpXError` code is `IPX_ERR_OK` and the function displays the video frame.
- Otherwise, the video frame is not displayed.

5.3.2.8 DisplayImage()

```
virtual IpXError IpXDisplay::DisplayImage (  
    IpXImage * image,  
    const char * mode = "auto" ) [pure virtual]
```

This function displays the still image. [QT: no].

Parameters

in	<i>image</i>	source image
in	<i>mode</i>	Display mode ("GDI", "OpenGL" mode or "auto" (default) for auto-detection)

Returns

Returns an error code:

- If successful, the `IpXError` code is `IPX_ERR_OK` and the function displays the still image.
- Otherwise, the still image is not displayed.

5.3.2.9 ConvertImage()

```
virtual IpXError IpXDisplay::ConvertImage (  
    IpXImage * source,  
    IpXImage * output ) [pure virtual]
```

This function converts the source image to the specified output image. [QT: no].

Parameters

in	<i>source</i>	source image
----	---------------	--------------

The only input source Pixel Types supported are shown in the tables below:

Table 5.27 BAYER CFA Pixel Types

Bayer Pattern Filter	8-bit	10-bit	12-bit	14-bit	16-bit
GR Pattern Filter	II_PIX_BAYGR8	II_PIX_BAYG↔ R10	II_PIX_BAYG↔ R12	II_PIX_BAYG↔ R14	II_PIX_BAYG↔ R16
RG Pattern Filter	II_PIX_BAYRG8	II_PIX_BAYR↔ G10	II_PIX_BAYR↔ G12	II_PIX_BAYR↔ G14	II_PIX_BAYR↔ G16
BG Pattern Filter	II_PIX_BAYBG8	II_PIX_BAYB↔ G10	II_PIX_BAYB↔ G12	II_PIX_BAYB↔ G14	II_PIX_BAYB↔ G16
GB Pattern Filter	II_PIX_BAYGB8	II_PIX_BAYG↔ B10	II_PIX_BAYG↔ B12	II_PIX_BAYG↔ B14	II_PIX_BAYG↔ B16

Table 5.28 PACKED BAYER CFA Pixel Types

Bayer Pattern Filter	10-bit	12-bit
GR Pattern Filter	II_PIX_BAYGR10_PACKED	II_PIX_BAYGR12_PACKED
RG Pattern Filter	II_PIX_BAYRG10_PACKED	II_PIX_BAYRG12_PACKED
GB Pattern Filter	II_PIX_BAYGB10_PACKED	II_PIX_BAYGB12_PACKED
GB Pattern Filter	II_PIX_BAYBG10_PACKED	II_PIX_BAYBG12_PACKED

Table 5.29 TrueSense CFA Pixel Types

TS Pattern Filter	8-bit	10-bit	12-bit	14-bit
BGGR w/ WBBW0	II_PIX_TS_BGGR↔ _WBBW0_8	II_PIX_TS_BGGR↔ _WBBW0_10	II_PIX_TS_BGGR↔ _WBBW0_12	II_PIX_TS_BGGR↔ _WBBW0_14
BGGR w/ WBBW1	II_PIX_TS_BGGR↔ _WBBW1_8	II_PIX_TS_BGGR↔ _WBBW1_10	II_PIX_TS_BGGR↔ _WBBW1_12	II_PIX_TS_BGGR↔ _WBBW1_14
BGGR w/ WBBW2	II_PIX_TS_BGGR↔ _WBBW2_8	II_PIX_TS_BGGR↔ _WBBW2_10	II_PIX_TS_BGGR↔ _WBBW2_12	II_PIX_TS_BGGR↔ _WBBW2_14
BGGR w/ WBBW3	II_PIX_TS_BGGR↔ _WBBW3_8	II_PIX_TS_BGGR↔ _WBBW3_10	II_PIX_TS_BGGR↔ _WBBW3_12	II_PIX_TS_BGGR↔ _WBBW3_14
GBRG w/ WGGW0	II_PIX_TS_GBRG↔ _WGGW0_8	II_PIX_TS_GBRG↔ _WGGW0_10	II_PIX_TS_GBRG↔ _WGGW0_12	II_PIX_TS_GBRG↔ _WGGW0_14
GBRG w/ WGGW1	II_PIX_TS_GBRG↔ _WGGW1_8	II_PIX_TS_GBRG↔ _WGGW1_10	II_PIX_TS_GBRG↔ _WGGW1_12	II_PIX_TS_GBRG↔ _WGGW1_14
GBRG w/ WGGW2	II_PIX_TS_GBRG↔ _WGGW2_8	II_PIX_TS_GBRG↔ _WGGW2_10	II_PIX_TS_GBRG↔ _WGGW2_12	II_PIX_TS_GBRG↔ _WGGW2_14
GBRG w/ WGGW3	II_PIX_TS_GBRG↔ _WGGW3_8	II_PIX_TS_GBRG↔ _WGGW3_10	II_PIX_TS_GBRG↔ _WGGW3_12	II_PIX_TS_GBRG↔ _WGGW3_14
GRBG w/ WGGW0	II_PIX_TS_GRBG↔ _WGGW0_8	II_PIX_TS_GRBG↔ _WGGW0_10	II_PIX_TS_GRBG↔ _WGGW0_12	II_PIX_TS_GRBG↔ _WGGW0_14
GRBG w/ WGGW1	II_PIX_TS_GRBG↔ _WGGW1_8	II_PIX_TS_GRBG↔ _WGGW1_10	II_PIX_TS_GRBG↔ _WGGW1_12	II_PIX_TS_GRBG↔ _WGGW1_14
GRBG w/ WGGW2	II_PIX_TS_GRBG↔ _WGGW2_8	II_PIX_TS_GRBG↔ _WGGW2_10	II_PIX_TS_GRBG↔ _WGGW2_12	II_PIX_TS_GRBG↔ _WGGW2_14
GRBG w/ WGGW3	II_PIX_TS_GRBG↔ _WGGW3_8	II_PIX_TS_GRBG↔ _WGGW3_10	II_PIX_TS_GRBG↔ _WGGW3_12	II_PIX_TS_GRBG↔ _WGGW3_14

TS Pattern Filter	8-bit	10-bit	12-bit	14-bit
RGGB w/ WRRW0	II_PIX_TS_RGGB↔ _WRRW0_8	II_PIX_TS_RGGB↔ _WRRW0_10	II_PIX_TS_RGGB↔ _WRRW0_12	II_PIX_TS_RGGB↔ _WRRW0_14
RGGB w/ WRRW1	II_PIX_TS_RGGB↔ _WRRW1_8	II_PIX_TS_RGGB↔ _WRRW1_10	II_PIX_TS_RGGB↔ _WRRW1_12	II_PIX_TS_RGGB↔ _WRRW1_14
RGGB w/ WRRW2	II_PIX_TS_RGGB↔ _WRRW2_8	II_PIX_TS_RGGB↔ _WRRW2_10	II_PIX_TS_RGGB↔ _WRRW2_12	II_PIX_TS_RGGB↔ _WRRW2_14
RGGB w/ WRRW3	II_PIX_TS_RGGB↔ _WRRW3_8	II_PIX_TS_RGGB↔ _WRRW3_10	II_PIX_TS_RGGB↔ _WRRW3_12	II_PIX_TS_RGGB↔ _WRRW3_14

Parameters

<i>in</i>	<i>output</i>	destination image
-----------	---------------	-------------------

Returns

Returns an error code:

- If successful, the `IpxError` code is `IPX_ERR_OK` and the function converts the image.
- Otherwise, the source image is not converted.

5.3.2.10 Translate()

```
virtual IpxError IpxDisplay::Translate (
    int32_t * x,
    int32_t * y,
    int32_t flags ) [inline], [virtual]
```

This function translates the display object to the specified coordinates as indicated by the flag. [QT: yes].

Parameters

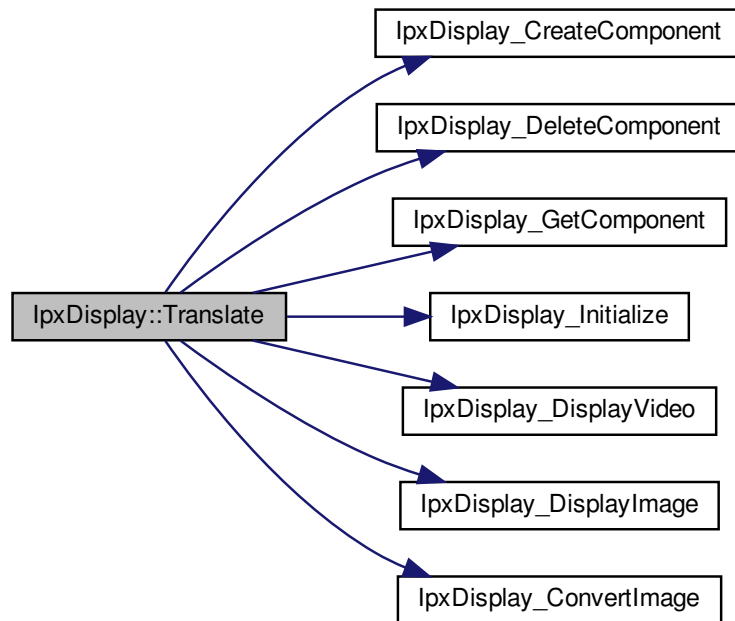
<i>in</i>	<i>x</i>	x-coordinate position
<i>in</i>	<i>y</i>	y-coordinate position
<i>in</i>	<i>flags</i>	Flag indicating mode to translate coordinates <ul style="list-style-type: none"> • IDFL_SCR_IMG Translate coordinates from screen to image coordinates • IDFL_IMG_SCR Translate coordinates from image to screen coordinates

Returns

Returns an error code:

- If successful, the `IpxError` code is `IPX_ERR_OK` and the function translates the object.
- Otherwise, the object failed to translate

Here is the call graph for this function:



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

- `IpxDisplay.h`

5.4 IpxImage Struct Reference

Data structure for description of Imperx Image.

```
#include <IpxImage.h>
```

Public Attributes

- [uint32_t nSize](#)
- [uint32_t version](#)
- [lpxPixelTypeDescr pixelTypeDescr](#)
- [int32_t origin](#)
- [uint32_t width](#)
- [uint32_t height](#)
- [uint32_t imageSize](#)
- [uint32_t rowSize](#)
- [uint64_t timestamp](#)
- [uint64_t imageID](#)
- [lpxUserData * userData](#)
- [char * imageData](#)
- [char * imageDataOrigin](#)

5.4.1 Detailed Description

Data structure for description of Imperx Image.

Note

[lpxImage](#) stores the image data in a char array. The number of bytes per pixel channel is defined by Pixel Type. The field 'imageDataOrigin' is used by [lpxImage](#) API for memory management and should not be changed by user.

See also

[lpxPixelTypeDescr](#)
[II_PIXEL_TYPE_DEFINES](#)
[lpxCreateImage](#)
[lpxCreateImageHeader](#)
[lpxReleaseImage](#)

5.4.2 Member Data Documentation

5.4.2.1 nSize

```
uint32_t IpxImage::nSize
```

Size of the [lpxImage](#) structure

5.4.2.2 version

```
uint32_t IpxImage::version
```

Version of data structure for image.

5.4.2.3 pixelTypeDescr

```
IpxPixelTypeDescr IpxImage::pixelTypeDescr
```

Descriptor of pixel format for image.

5.4.2.4 origin

```
int32_t IpxImage::origin
```

Origin of Image coordinate system: 0 - top-left origin: IIPL_ORIGIN_TL; 1 - bottom-left origin: IIPL_ORIGIN_BL (Windows bitmaps style).

5.4.2.5 width

```
uint32_t IpxImage::width
```

Image width in pixels.

5.4.2.6 height

```
uint32_t IpxImage::height
```

Image height in pixels.

5.4.2.7 imageSize

```
uint32_t IpxImage::imageSize
```

Image data size in bytes (equals image->height*image->rowSize in case of interleaved data).

5.4.2.8 rowSize

```
uint32_t IpxImage::rowSize
```

Size of aligned image row in bytes (not necessarily aligned) - needed for correct deallocation.

5.4.2.9 timestamp

```
uint64_t IpxImage::timestamp
```

Image timestamp.

5.4.2.10 imageID

```
uint64_t IpxImage::imageID
```

Image identifier. For U3V and GEV - block_id field of Leader packet

5.4.2.11 userData

```
IpxUserData* IpxImage::userData
```

User data, linked with this image.

5.4.2.12 imageData

```
char* IpxImage::imageData
```

Pointer to aligned image data.

5.4.2.13 imageDataOrigin

```
char* IpxImage::imageDataOrigin
```

Pointer to very origin of image data.

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

- `lpxImage.h`

5.5 lpxImageConverter Class Reference

A Class for [lpxImageConverter](#) modules that contains methods to convert [lpxImage](#) images.

```
#include <IpxImageConverter.h>
```


Public Member Functions

- virtual [IpxComponent](#) * [GetComponent](#) ()=0
This function returns the pointer to the [IpxComponent](#) object.
- virtual [IpxError](#) [ConvertImage](#) ([IpxImage](#) *source, [IpxImage](#) *output)=0
This function converts the input source [IpxImage](#) to the targeted output destination [IpxImage](#).
- virtual [IpxError](#) [IICovert](#) ([IpxImage](#) *image_in, unsigned long outPixelFormat, [IpxImage](#) **image_out)=0
This function converts the input source [IpxImage](#) to the targeted output destination [IpxImage](#) based on the output pixel type.

Static Public Member Functions

- static IPXC_API [IpxImageConverter](#) * [CreateComponent](#) ()
This function returns the created [IpxImageConverter](#) instance.
- static IPXC_API void [DeleteComponent](#) ([IpxImageConverter](#) *component)
This function deletes the [IpxImageConverter](#) component and all associated resources obtained by the [IpxImageConverter](#) object.

5.5.1 Detailed Description

A Class for [IpxImageConverter](#) modules that contains methods to convert [IpxImage](#) images.

A class containing methods for [IpxImageConverter](#) modules.

5.5.2 Member Function Documentation

5.5.2.1 CreateComponent()

```
static IPXC_API IpxImageConverter* IpxImageConverter::CreateComponent ( ) [static]
```

This function returns the created [IpxImageConverter](#) instance.

Returns

Returns the created [IpxImageConverter](#) object

5.5.2.2 DeleteComponent()

```
static IPXC_API void IpxImageConverter::DeleteComponent (
    IpxImageConverter * component ) [static]
```

This function deletes the [IpxImageConverter](#) component and all associated resources obtained by the [IpxImageConverter](#) object.

Parameters

in	<i>component</i>	Pointer to the IpxImageConverter object
----	------------------	---

Returns

Returns void

5.5.2.3 GetComponent()

```
virtual IpxComponent* IpxImageConverter::GetComponent ( ) [pure virtual]
```

This function returns the pointer to the [IpxComponent](#) object.

Returns

Returns the Pointer to the [IpxComponent](#) object

5.5.2.4 ConvertImage()

```
virtual IpxError IpxImageConverter::ConvertImage (
    IpxImage * source,
    IpxImage * output ) [pure virtual]
```

This function converts the input source [IpxImage](#) to the targeted output destination [IpxImage](#).

Parameters

in	<i>source</i>	Pointer to the input source IpxImage
out	<i>output</i>	Pointer to the output destination IpxImage

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully converts the source [IpxImage](#) to the targeted output destination [IpxImage](#)
- If `IpxError` error code < 0, then it returns a negative error code indicating problems converting the [IpxImage](#)

5.5.2.5 IConvert()

```
virtual IpxError IpxImageConverter::IConvert (
    IpxImage * image_in,
    unsigned long outPixelFormat,
    IpxImage ** image_out ) [pure virtual]
```

This function converts the input source [IpxImage](#) to the targeted output destination [IpxImage](#) based on the output pixel type.

Parameters

in	<i>image_in</i>	input source IpxImage image
in	<i>outPixelFormat</i>	Output pixel type
out	<i>image_out</i>	Pointer to the output destination IpxImage

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully converts the source [IpxImage](#) to the targeted output destination [IpxImage](#) based on the output pixel type.
- If `IpxError` error code < 0 , then it returns a negative error code indicating problems converting the [IpxImage](#)

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

- `IpxImageConverter.h`

5.6 IpxImageSerializer Class Reference

[IpxComponent](#) to save [IpxImage](#) to disk.

```
#include <IpxImageSerializer.h>
```

Public Member Functions

- virtual `IpxComponent * GetComponent ()=0`
This function returns the pointer to the [IpxComponent](#) object.
- virtual `IpxError StartSeriesRecord (IpxImage *pSrc, const char *format)=0`
This function starts the recording session for series of images of the same format.
- virtual `IpxError StartMovieRecord (IpxImage *pSrc, const char *fileName, double fps)=0`
This function starts the recording session for movies.
- virtual `IpxError FinishRecord ()=0`
This function ends the recording session.
- virtual `IpxError Save (IpxImage *image, const char *fileName=0)=0`
- virtual `IpxError Load (IpxImage *image, const char *fileName)=0`
This function reads and loads the standalone image record.
- virtual `IpxError GetImageHeader (IpxImage *image, const char *fileName)=0`
This function reads and loads the standalone image header.
- virtual `IpxError Free (IpxImage *image)=0`
This function frees the image loaded with [IpxImageSerializer](#).

Static Public Member Functions

- static IPXS_API [IpxImageSerializer](#) * [CreateComponent](#) (bool enableMovies=true)
This function returns the IpxHandle for the created [IpxImageSerializer](#) instance.
- static IPXS_API void [DeleteComponent](#) ([IpxImageSerializer](#) *component)
This function returns the IpxHandle for the created [IpxImageSerializer](#) instance.

5.6.1 Detailed Description

[IpxComponent](#) to save [IpxImage](#) to disk.

[IpxImageSerializer](#)

5.6.2 Member Function Documentation

5.6.2.1 CreateComponent()

```
static IPXS_API IpxImageSerializer* IpxImageSerializer::CreateComponent (
    bool enableMovies = true ) [static]
```

This function returns the IpxHandle for the created [IpxImageSerializer](#) instance.

Parameters

in	<i>enableMovies</i>	flag to enable Movies
----	---------------------	-----------------------

Returns

Returns the IpxHandle for the created [IpxImageSerializer](#) object

5.6.2.2 DeleteComponent()

```
static IPXS_API void IpxImageSerializer::DeleteComponent (
    IpxImageSerializer * component ) [static]
```

This function returns the IpxHandle for the created [IpxImageSerializer](#) instance.

Parameters

in	<i>component</i>	Pointer to IpxImageSerializer component
----	------------------	---

Returns

void

5.6.2.3 GetComponent()

```
virtual IpxComponent* IpxImageSerializer::GetComponent ( ) [pure virtual]
```

This function returns the pointer to the [IpxComponent](#) object.

The [IpxComponent](#) object will give access to the data member functions shown below:

- 🔒 GetComponentTypeID
- 🔒 GetParamAsString
- 🔒 GetParamBool
- 🔒 GetParamCount
- 🔒 GetParamFloat
- 🔒 GetParamInt
- 🔒 GetParamName
- 🔒 GetParamString
- 🔒 RunCommand
- 🔒 SetParamAsString
- 🔒 SetParamBool
- 🔒 SetParamFloat
- 🔒 SetParamInt
- 🔒 SetParamString
- 🔒 ~IpxComponent

Returns

Returns the Pointer to the [IpxComponent](#) object

The following example will illustrate on how to access the [IpxComponent](#) data member function:

```
IpxImageSerializer* serializer =
    IpxImageSerializer::CreateComponent();
IpxError res = IPX_ERR_UNKNOWN;

if (serializer) {
    (serializer->GetComponent())->SetParamInt(ISP_JPEG_QUALITY, mJpgQuality);
    res = serializer->Save(image, fileName);
}

...

IpxImageSerializer::DeleteComponent(serializer);
```

5.6.2.4 StartSeriesRecord()

```
virtual IpxError IpxImageSerializer::StartSeriesRecord (
    IpxImage * pSrc,
    const char * format ) [pure virtual]
```

This function starts the recording session for series of images of the same format.

Parameters

in	<i>pSrc</i>	input source Imperx Image
in	<i>format</i>	Image Format Type

Returns

Returns the error code:

- IPX_ERR_OK Successfully starts the recording series.
- If IpxError error code < 0, then it returns a negative error code indicating problem starting the recording series.

5.6.2.5 StartMovieRecord()

```
virtual IpxError IpxImageSerializer::StartMovieRecord (
    IpxImage * pSrc,
    const char * fileName,
    double fps ) [pure virtual]
```

This function starts the recording session for movies.

Parameters

in	<i>pSrc</i>	input source Imperx Image
in	<i>fileName</i>	file name
in	<i>fps</i>	frames per second

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully starts the recording session for movies.
- If `IpxError` error code < 0 , then it returns a negative error code indicating problem starting the recording session for movie

5.6.2.6 FinishRecord()

```
virtual IpxError IpxImageSerializer::FinishRecord ( ) [pure virtual]
```

This function ends the recording session.

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully ends the recording session.
- If `IpxError` error code < 0 , then it returns a negative error code indicating problem ending the recording session

5.6.2.7 Save()

```
virtual IpxError IpxImageSerializer::Save (
    IpxImage * image,
    const char * fileName = 0 ) [pure virtual]
```

This function saves the standalone image or puts the image to recording session if [StartSeriesRecord\(\)](#) or [StartMovieRecord\(\)](#) method was called

Parameters

in	<i>image</i>	input source Imperx Image
in	<i>fileName</i>	file name

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully saves the standalone image
- If `lpxError` error code < 0 , then it returns a negative error code indicating problem saving the record

5.6.2.8 Load()

```
virtual IpxError IpxImageSerializer::Load (  
    IpxImage * image,  
    const char * fileName ) [pure virtual]
```

This function reads and loads the standalone image record.

Parameters

in	<i>image</i>	input source Imperx Image
in	<i>fileName</i>	file name

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully allocates the [lpxImage](#) data.
- If `lpxError` error code < 0 , then it returns a negative error code indicating problems reading the standalone image

5.6.2.9 GetImageHeader()

```
virtual IpxError IpxImageSerializer::GetImageHeader (  
    IpxImage * image,  
    const char * fileName ) [pure virtual]
```

This function reads and loads the standalone image header.

Parameters

in	<i>image</i>	input source Imperx Image
in	<i>fileName</i>	file name(wide char)

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully allocates the [IpxImage](#) data.
- If `IpxError` error code < 0 , then it returns a negative error code indicating problems reading the standalone image

5.6.2.10 Free()

```
virtual IpxError IpxImageSerializer::Free (
    IpxImage * image ) [pure virtual]
```

This function frees the image loaded with [IpxImageSerializer](#).

Parameters

in	<i>image</i>	input source Imperx Image
----	--------------	---------------------------

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully frees the allocates memory of the [IpxImage](#) image.
- If `IpxError` error code < 0 , then it returns a negative error code indicating problems freeing the loaded image

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

- [IpxImageSerializer.h](#)

5.7 IpxImageUnpacker Class Reference

[IpxComponent](#) to unpack images.

```
#include <IpxImageUnpacker.h>
```

Public Member Functions

- virtual [IpxComponent](#) * [GetComponent](#) ()=0
This function returns the pointer to the [IpxComponent](#) object.
- virtual [IpxError](#) [Unpack](#) ([IpxImage](#) *source, [IpxImage](#) *output, void *ptr=0)=0
This function transforms the packed source image to the unpacked one.

Static Public Member Functions

- static IPXU_API [IpxImageUnpacker](#) * [CreateComponent](#) ()
This function returns the created [IpxImageUnpacker](#) instance.
- static IPXU_API void [DeleteComponent](#) ([IpxImageUnpacker](#) *component)
This function deletes the [IpxImageUnpacker](#) component and all associated resources obtained by the [IpxImageUnpacker](#) object.

5.7.1 Detailed Description

[IpxComponent](#) to unpack images.

[IpxImageUnpacker](#)

5.7.2 Member Function Documentation

5.7.2.1 CreateComponent()

```
static IPXU_API IpxImageUnpacker* IpxImageUnpacker::CreateComponent ( ) [static]
```

This function returns the created [IpxImageUnpacker](#) instance.

Returns

Returns the created [IpxImageUnpacker](#) object

5.7.2.2 DeleteComponent()

```
static IPXU_API void IpxImageUnpacker::DeleteComponent (
    IpxImageUnpacker * component ) [static]
```

This function deletes the [IpxImageUnpacker](#) component and all associated resources obtained by the [IpxImageUnpacker](#) object.

Parameters

in	<i>component</i>	Pointer to the IpxImageUnpacker object
----	------------------	--

Returns

Returns void

5.7.2.3 GetComponent()

```
virtual IpxComponent* IpxImageUnpacker::GetComponent ( ) [pure virtual]
```

This function returns the pointer to the [IpxComponent](#) object.

Returns

Returns the Pointer to the [IpxComponent](#) object

5.7.2.4 Unpack()

```
virtual IpxError IpxImageUnpacker::Unpack (
    IpxImage * source,
    IpxImage * output,
    void * ptr = 0 ) [pure virtual]
```

This function transforms the packed source image to the unpacked one.

Parameters

<i>source</i>	Pointer to source IpxImage .
<i>output</i>	Pointer to destination IpxImage .
<i>ptr</i>	Pointer to private data.

Returns

If the function succeeds, the return value is 0. If the function fails, the return value is non-zero.

Note

This function transforms the source RAW image to the destination image by unpacking the image and deinterlacing if necessary. Also, it allocates the destination output image memory if the user didn't pre-allocation the destination image.

For example:

```
IpxImageUnpacker*  unpacker = IpxImageUnpacker::CreateComponent
                        ();
IpxError           res = IPX_ERR_UNKNOWN;

...

if (unpacker) res = unpacker->Unpack(source, output);

if (!IPX_ERR_SUCCEEDED(res)) {
    //TODO: Process error
}

IpxImageUnpacker::DeleteComponent(unpacker);
```

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

- [IpxImageUnpacker.h](#)

5.8 IpxImgProcessor Class Reference

Pure virtual base class for image processor.

```
#include <IpxImgProcessor.h>
```

5.8.1 Detailed Description

Pure virtual base class for image processor.

[IpxImgProcessor](#)

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

- [IpxImgProcessor.h](#)

5.9 IpxPixelFormatDescr Struct Reference

Base type of data for description of [IpxImage](#) and other image data types.

```
#include <IpxPixelFormat.h>
```

Public Attributes

- [uint32_t pixelType](#)
- [uint32_t depth](#)
- [bool pixSigned](#)
- [uint32_t pixAlign](#)
- [uint32_t channels](#)
- [uint32_t pixSize](#)

5.9.1 Detailed Description

Base type of data for description of [IpxImage](#) and other image data types.

Data structure for description of pixel format.

Note

[IpxPixelFormatDescr](#) stores parameters of the pixel format.

See also

[II_PIXEL_TYPE_DEFINES](#)
[IpxCreateImage](#)
[IpxCreateImageHeader](#)
[IpxReleaseImage](#)

5.9.2 Member Data Documentation

5.9.2.1 pixelType

```
uint32_t IpxPixelFormatDescr::pixelType
```

Pixel type.

5.9.2.2 depth

```
uint32_t IpxPixelFormatDescr::depth
```

Bit depth of channels.

5.9.2.3 pixSigned

```
bool IpxPixelFormatDescr::pixSigned
```

true for signed pixel.

5.9.2.4 pixAlign

```
uint32_t IpxPixelFormatDescr::pixAlign
```

Pixel packing (packed/normalized).

5.9.2.5 channels

```
uint32_t IpxPixelTypeDescr::channels
```

Number of channels.

5.9.2.6 pixSize

```
uint32_t IpxPixelTypeDescr::pixSize
```

Pixel size in bits.

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

- `IpxPixelType.h`

5.10 IpxPoint Struct Reference

The [IpxPoint](#) structure specifies a point.

```
#include <IpxToolsBase.h>
```

Public Attributes

- `int x`
- `int y`

5.10.1 Detailed Description

The [IpxPoint](#) structure specifies a point.

5.10.2 Member Data Documentation

5.10.2.1 x

```
int IpxPoint::x
```

Specifies the x coordinate of the point.

5.10.2.2 y

```
int IpxPoint::y
```

Specifies the y coordinate of the point.

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

- IpxToolsBase.h

5.11 IpxRect Struct Reference

The [IpxRect](#) structure defines a rectangle by the coordinates of its upper-left corner and width, height.

```
#include <IpxToolsBase.h>
```

Public Attributes

- int [x](#)
- int [y](#)
- int [width](#)
- int [height](#)

5.11.1 Detailed Description

The [IpxRect](#) structure defines a rectangle by the coordinates of its upper-left corner and width, height.

5.11.2 Member Data Documentation

5.11.2.1 x

```
int IpxRect::x
```

Specifies the x-coordinate of the upper-left corner of the rectangle.

5.11.2.2 y

```
int IpxRect::y
```

Specifies the y-coordinate of the upper-left corner of the rectangle.

5.11.2.3 width

```
int IpxRect::width
```

Specifies the width of the rectangle.

5.11.2.4 height

```
int IpxRect::height
```

Specifies the height of the rectangle.

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

- `IpxToolsBase.h`

5.12 IpxSize Struct Reference

The [IpxSize](#) structure specifies a rectangle.

```
#include <IpxToolsBase.h>
```

Public Attributes

- int [width](#)
- int [height](#)

5.12.1 Detailed Description

The [IpxSize](#) structure specifies a rectangle.

5.12.2 Member Data Documentation

5.12.2.1 width

```
int IpxSize::width
```

Specifies the width of the rectangle.

5.12.2.2 height

```
int IpxSize::height
```

Specifies the height of the rectangle.

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

- lpxToolsBase.h

5.13 lpxTrueSense Class Reference

A Class for [lpxTrueSense](#) modules that contains methods to convert [lpxImage](#) images.

```
#include <IpxTrueSense.h>
```

Public Member Functions

- virtual [lpxComponent](#) * [GetComponent](#) ()=0
This function returns the pointer to the [lpxComponent](#) object.
- virtual [lpxError](#) [ConvertImage](#) (const [lpxImage](#) *pSrc, [lpxImage](#) *pDst)=0
This TrueSense CFA Demosaicing function converts the input source [lpxImage](#) to the targeted output destination [lpxImage](#).
- virtual [lpxError](#) [AllocData](#) (const [lpxImage](#) *pSrc, [lpxImage](#) *pDst)=0
This function allocates memory.
- virtual void [ReleaseData](#) ()=0
This function releases the allocated memory.

Static Public Member Functions

- static TS_API [lpxTrueSense](#) * [CreateComponent](#) ()
This function returns the created [lpxTrueSense](#) instance.
- static TS_API void [DeleteComponent](#) ([lpxTrueSense](#) *in)
This function deletes the [lpxTrueSense](#) component and all associated resources obtained by the [lpxTrueSense](#) object.

5.13.1 Detailed Description

A Class for [lpxTrueSense](#) modules that contains methods to convert [lpxImage](#) images.

A class containing methods for [lpxTrueSense](#) modules.

5.13.2 Member Function Documentation

5.13.2.1 CreateComponent()

```
static TS_API IpxTrueSense* IpxTrueSense::CreateComponent ( ) [static]
```

This function returns the created [IpxTrueSense](#) instance.

Returns

Returns the created [IpxTrueSense](#) object

5.13.2.2 DeleteComponent()

```
static TS_API void IpxTrueSense::DeleteComponent (
    IpxTrueSense * in ) [static]
```

This function deletes the [IpxTrueSense](#) component and all associated resources obtained by the [IpxTrueSense](#) object.

Parameters

in	in	Pointer to the IpxTrueSense object
----	----	--

Returns

Returns void

5.13.2.3 GetComponent()

```
virtual IpxComponent* IpxTrueSense::GetComponent ( ) [pure virtual]
```

This function returns the pointer to the [IpxComponent](#) object.

The [IpxComponent](#) object will give access to the data member functions shown below:

- 🔗 GetComponentTypeID
- 🔗 GetParamAsString
- 🔗 GetParamBool
- 🔗 GetParamCount
- 🔗 GetParamFloat
- 🔗 GetParamInt
- 🔗 GetParamName
- 🔗 GetParamString
- 🔗 RunCommand
- 🔗 SetParamAsString
- 🔗 SetParamBool
- 🔗 SetParamFloat
- 🔗 SetParamInt
- 🔗 SetParamString
- 🔗 ~IpxComponent

Returns

Returns the Pointer to the [IpxComponent](#) object

The following example will illustrate on how to access the [IpxComponent](#) data member function:

```
IpxTrueSense *pDeTS = IpxTrueSense::CreateComponent();
IpxError err = pDeTS->GetComponent()->SetParamInt("TrueSenseAlgType", 3);
pDeTS->ConvertImage(imageIN, imageOUT);
IpxTrueSense::DeleteComponent(pDeTS);
```

5.13.2.4 ConvertImage()

```
virtual IpxError IpxTrueSense::ConvertImage (
    const IpxImage * pSrc,
    IpxImage * pDst ) [pure virtual]
```

This TrueSense CFA Demosaicing function converts the input source [IpxImage](#) to the targeted output destination [IpxImage](#).

Parameters

in	pSrc	Pointer to the input source lpxImage
----	------	--

The only input source Pixel Types supported are shown in the tables below:

Table 5.47 TrueSense CFA Pixel Types

TS Pattern Filter	8-bit	10-bit	12-bit	14-bit
BGGR w/ WBBW0	II_PIX_TS_BGGR↔ _WBBW0_8	II_PIX_TS_BGGR↔ _WBBW0_10	II_PIX_TS_BGGR↔ _WBBW0_12	II_PIX_TS_BGGR↔ _WBBW0_14
BGGR w/ WBBW1	II_PIX_TS_BGGR↔ _WBBW1_8	II_PIX_TS_BGGR↔ _WBBW1_10	II_PIX_TS_BGGR↔ _WBBW1_12	II_PIX_TS_BGGR↔ _WBBW1_14
BGGR w/ WBBW2	II_PIX_TS_BGGR↔ _WBBW2_8	II_PIX_TS_BGGR↔ _WBBW2_10	II_PIX_TS_BGGR↔ _WBBW2_12	II_PIX_TS_BGGR↔ _WBBW2_14
BGGR w/ WBBW3	II_PIX_TS_BGGR↔ _WBBW3_8	II_PIX_TS_BGGR↔ _WBBW3_10	II_PIX_TS_BGGR↔ _WBBW3_12	II_PIX_TS_BGGR↔ _WBBW3_14
GBRG w/ WGGW0	II_PIX_TS_GBRG↔ _WGGW0_8	II_PIX_TS_GBRG↔ _WGGW0_10	II_PIX_TS_GBRG↔ _WGGW0_12	II_PIX_TS_GBRG↔ _WGGW0_14
GBRG w/ WGGW1	II_PIX_TS_GBRG↔ _WGGW1_8	II_PIX_TS_GBRG↔ _WGGW1_10	II_PIX_TS_GBRG↔ _WGGW1_12	II_PIX_TS_GBRG↔ _WGGW1_14
GBRG w/ WGGW2	II_PIX_TS_GBRG↔ _WGGW2_8	II_PIX_TS_GBRG↔ _WGGW2_10	II_PIX_TS_GBRG↔ _WGGW2_12	II_PIX_TS_GBRG↔ _WGGW2_14
GBRG w/ WGGW3	II_PIX_TS_GBRG↔ _WGGW3_8	II_PIX_TS_GBRG↔ _WGGW3_10	II_PIX_TS_GBRG↔ _WGGW3_12	II_PIX_TS_GBRG↔ _WGGW3_14
GRBG w/ WGGW0	II_PIX_TS_GRBG↔ _WGGW0_8	II_PIX_TS_GRBG↔ _WGGW0_10	II_PIX_TS_GRBG↔ _WGGW0_12	II_PIX_TS_GRBG↔ _WGGW0_14
GRBG w/ WGGW1	II_PIX_TS_GRBG↔ _WGGW1_8	II_PIX_TS_GRBG↔ _WGGW1_10	II_PIX_TS_GRBG↔ _WGGW1_12	II_PIX_TS_GRBG↔ _WGGW1_14
GRBG w/ WGGW2	II_PIX_TS_GRBG↔ _WGGW2_8	II_PIX_TS_GRBG↔ _WGGW2_10	II_PIX_TS_GRBG↔ _WGGW2_12	II_PIX_TS_GRBG↔ _WGGW2_14
GRBG w/ WGGW3	II_PIX_TS_GRBG↔ _WGGW3_8	II_PIX_TS_GRBG↔ _WGGW3_10	II_PIX_TS_GRBG↔ _WGGW3_12	II_PIX_TS_GRBG↔ _WGGW3_14
RGGB w/ WRRW0	II_PIX_TS_RGGB↔ _WRRW0_8	II_PIX_TS_RGGB↔ _WRRW0_10	II_PIX_TS_RGGB↔ _WRRW0_12	II_PIX_TS_RGGB↔ _WRRW0_14
RGGB w/ WRRW1	II_PIX_TS_RGGB↔ _WRRW1_8	II_PIX_TS_RGGB↔ _WRRW1_10	II_PIX_TS_RGGB↔ _WRRW1_12	II_PIX_TS_RGGB↔ _WRRW1_14
RGGB w/ WRRW2	II_PIX_TS_RGGB↔ _WRRW2_8	II_PIX_TS_RGGB↔ _WRRW2_10	II_PIX_TS_RGGB↔ _WRRW2_12	II_PIX_TS_RGGB↔ _WRRW2_14
RGGB w/ WRRW3	II_PIX_TS_RGGB↔ _WRRW3_8	II_PIX_TS_RGGB↔ _WRRW3_10	II_PIX_TS_RGGB↔ _WRRW3_12	II_PIX_TS_RGGB↔ _WRRW3_14

Parameters

out	pDst	Pointer to the output destination lpxImage
-----	------	--

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully converts the source [IpxImage](#) to the targeted output destination [IpxImage](#)
- If `IpxError` error code < 0 , then it returns a negative error code indicating problems converting the [IpxImage](#)

5.13.2.5 AllocData()

```
virtual IpxError IpxTrueSense::AllocData (
    const IpxImage * pSrc,
    IpxImage * pDst ) [pure virtual]
```

This function allocates memory.

Parameters

in	<i>pSrc</i>	Pointer to the input source IpxImage
in	<i>pDst</i>	Pointer to the output destination IpxImage

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully allocates data
- If `IpxError` error code < 0 , then it returns a negative error code indicating problem allocating memory

5.13.2.6 ReleaseData()

```
virtual void IpxTrueSense::ReleaseData ( ) [pure virtual]
```

This function releases the allocated memory.

Returns

Returns the error code:

- `IPX_ERR_OK` Successfully releases the allocated data
- If `IpxError` error code < 0 , then it returns a negative error code indicating problem releasing the data allocated

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

- `IpxTrueSense.h`

5.14 IpxUserData Struct Reference

Data structure for description of User Data linked with Imperx Image.

```
#include <IpxUserData.h>
```

Public Attributes

- unsigned long [type](#)
- unsigned long [id](#)
- unsigned long [size](#)
- void * [data](#)
- bool [createdIpx](#)
- [_IpxUserData](#) * [pNext](#)

5.14.1 Detailed Description

Data structure for description of User Data linked with Imperx Image.

Note

IpxTools Library provides only base operation for handling of user data.

See also

[IpxCreateUserData](#)
[IpxReleaseUserData](#)

5.14.2 Member Data Documentation

5.14.2.1 type

```
unsigned long IpxUserData::type
```

type Type of user data. (II_NOT_DATA, II_HASHTABLE, II_XML_DATA, II_CUSTOM_DATA)

5.14.2.2 id

```
unsigned long IpxUserData::id
```

id ID of user data (must be > 0).

5.14.2.3 size

```
unsigned long IpxUserData::size
```

size Size of user data.

5.14.2.4 data

```
void* IpxUserData::data
```

data User data.

5.14.2.5 createdIpx

```
bool IpxUserData::createdIpx
```

createdIpx Indicates if user data was created by IpxTools. If true, then the user data is created by IpxTools.

5.14.2.6 pNext

```
_IpxUserData* IpxUserData::pNext
```

pNext Pointer to next User data block, or nullptr if next block does not exist.

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

- IpxUserData.h

Index

- ~lpxComponent
 - lpxComponent, [167](#)
- AllocData
 - lpxBayer, [165](#)
 - lpxTrueSense, [207](#)
- BAYER_EA
 - DeBayer Algorithms, [56](#)
- BAYER_GRADIENT
 - DeBayer Algorithms, [56](#)
- BAYER_OPENGL_MHC
 - DeBayer Algorithms, [56](#)
- BAYER_OPENGL_MMA
 - DeBayer Algorithms, [56](#)
- BAYER_SIMPLE
 - DeBayer Algorithms, [55](#)
- channels
 - lpxPixelTypeDescr, [199](#)
- Component Type IDs, [49](#)
- ConvertImage
 - lpxBayer, [163](#)
 - lpxDisplay, [180](#)
 - lpxImageConverter, [188](#)
 - lpxTrueSense, [205](#)
- CreateComponent
 - lpxBayer, [162](#)
 - lpxDisplay, [175](#)
 - lpxImageConverter, [187](#)
 - lpxImageSerializer, [190](#)
 - lpxImageUnpacker, [196](#)
 - lpxTrueSense, [204](#)
- createdlpx
 - lpxUserData, [209](#)
- DEBAYER_ALGO_TYPE
 - DeBayer Parameters, [53](#)
- DEBAYER_NOREALLOCOT
 - DeBayer Parameters, [53](#)
- data
 - lpxUserData, [209](#)
- DeBayer Algorithms, [55](#)
 - BAYER_EA, [56](#)
 - BAYER_GRADIENT, [56](#)
 - BAYER_OPENGL_MHC, [56](#)
 - BAYER_OPENGL_MMA, [56](#)
 - BAYER_SIMPLE, [55](#)
- DeBayer Parameters, [53](#)
 - DEBAYER_ALGO_TYPE, [53](#)
 - DEBAYER_NOREALLOCOT, [53](#)
- DeleteComponent
 - lpxBayer, [162](#)
 - lpxDisplay, [175](#)
 - lpxImageConverter, [187](#)
 - lpxImageSerializer, [190](#)
 - lpxImageUnpacker, [196](#)
 - lpxTrueSense, [204](#)
- depth
 - lpxPixelTypeDescr, [199](#)
- Display Component Parameters, [10](#)
- DisplayImage
 - lpxDisplay, [180](#)
- DisplayVideo
 - lpxDisplay, [179](#)
- Dump Rect Parameters, [81](#)
 - IDP_DUMP_COLOR, [82](#)
 - IDP_DUMP_H, [82](#)
 - IDP_DUMP_W, [82](#)
 - IDP_DUMP_X, [81](#)
 - IDP_DUMP_Y, [82](#)
- Error Codes, [48](#)
- FinishRecord
 - lpxImageSerializer, [193](#)
- Fit Modes and Mouse Processing, [92](#)
 - IPXD_FIT_FILL, [92](#)
 - IPXD_FIT_FULLSIZE, [93](#)
 - IPXD_FIT_NONE, [92](#)
 - IPXD_FIT_WINDOW, [92](#)
 - IPXD_MOUSE_DEFAULT, [93](#)
 - IPXD_MOUSE_LOCK, [93](#)
 - IPXD_MOUSE_SKIP, [93](#)
- Free
 - lpxImageSerializer, [195](#)
- GetComponent
 - lpxBayer, [162](#)
 - lpxDisplay, [176](#)
 - lpxImageConverter, [188](#)
 - lpxImageSerializer, [191](#)

- lpxImageUnpacker, [197](#)
 - lpxTrueSense, [204](#)
- GetComponentTypeID
 - lpxComponent, [167](#)
- GetImageHeader
 - lpxImageSerializer, [194](#)
- GetParamArray
 - lpxComponent, [173](#)
- GetParamAsString
 - lpxComponent, [168](#)
- GetParamBool
 - lpxComponent, [171](#)
- GetParamCount
 - lpxComponent, [167](#)
- GetParamFloat
 - lpxComponent, [172](#)
- GetParamInt
 - lpxComponent, [172](#)
- GetParamName
 - lpxComponent, [167](#)
- GetParamString
 - lpxComponent, [173](#)
- GetSystemInfo
 - lpxDisplay, [177](#)
- height
 - lpxImage, [185](#)
 - lpxRect, [202](#)
 - lpxSize, [202](#)
- IDFL_IMG_SCR
 - Translate Flags, [91](#)
- IDFL_SCR_IMG
 - Translate Flags, [91](#)
- IDP_BACKGROUND
 - Pre-initialization Parameters, [63](#)
- IDP_CALC_COEF_B
 - White Balance Correction Parameters, [77](#)
- IDP_CALC_COEF_G
 - White Balance Correction Parameters, [77](#)
- IDP_CALC_COEF_R
 - White Balance Correction Parameters, [76](#)
- IDP_COMMAND_WINDOW
 - Pre-initialization Parameters, [66](#)
- IDP_CORR_GAIN_B
 - Software Image Correction Parameters, [73](#)
- IDP_CORR_GAIN_G
 - Software Image Correction Parameters, [73](#)
- IDP_CORR_GAIN_R
 - Software Image Correction Parameters, [73](#)
- IDP_CORR_GAMMA
 - Software Image Correction Parameters, [75](#)
- IDP_CORR_MODE
 - Software Image Correction Parameters, [73](#)
- IDP_CORR_OFFSETS_B
 - Software Image Correction Parameters, [74](#)
- IDP_CORR_OFFSETS_G
 - Software Image Correction Parameters, [74](#)
- IDP_CORR_OFFSETS_R
 - Software Image Correction Parameters, [74](#)
- IDP_DUMP_COLOR
 - Dump Rect Parameters, [82](#)
- IDP_DUMP_H
 - Dump Rect Parameters, [82](#)
- IDP_DUMP_W
 - Dump Rect Parameters, [82](#)
- IDP_DUMP_X
 - Dump Rect Parameters, [81](#)
- IDP_DUMP_Y
 - Dump Rect Parameters, [82](#)
- IDP_GDI_BAYER
 - Pre-initialization Parameters, [65](#)
- IDP_GDI_TRUESENSE
 - Pre-initialization Parameters, [65](#)
- IDP_INIT_AT_X
 - Pre-initialization Parameters, [63](#)
- IDP_INIT_AT_Y
 - Pre-initialization Parameters, [64](#)
- IDP_INIT_FIT
 - Pre-initialization Parameters, [63](#)
- IDP_MANAGED_FPS
 - Run-time Parameters, [69](#)
- IDP_MANAGED_STATE
 - Run-time Parameters, [70](#)
- IDP_MENU_CMD
 - Run-time Parameters, [71](#)
- IDP_MENU_X
 - Run-time Parameters, [71](#)
- IDP_MENU_Y
 - Run-time Parameters, [71](#)
- IDP_OGL_BAYER
 - Pre-initialization Parameters, [64](#)
- IDP_OGL_TRUESENSE
 - Pre-initialization Parameters, [65](#)
- IDP_OVERLAY_BGMODE
 - Overlay Text Parameters, [79](#)
- IDP_OVERLAY_COLOR
 - Overlay Text Parameters, [79](#)
- IDP_OVERLAY_FONT_DESC_0
 - Pre-initialization Parameters, [66](#)
- IDP_OVERLAY_FONT_DESC_1
 - Pre-initialization Parameters, [66](#)
- IDP_OVERLAY_FONT_DESC_2
 - Pre-initialization Parameters, [67](#)
- IDP_OVERLAY_FONT_DESC_3
 - Pre-initialization Parameters, [67](#)
- IDP_OVERLAY_FONT
 - Overlay Text Parameters, [79](#)
- IDP_OVERLAY_INDEX

- Overlay Text Parameters, [78](#)
- IDP_OVERLAY_POS
 - Overlay Text Parameters, [79](#)
- IDP_OVERLAY_TEXT
 - Overlay Text Parameters, [80](#)
- IDP_PROC_PROCESSOR_TYPE
 - Run-time Parameters, [70](#)
- IDP_PROC_PROCESSOR
 - Run-time Parameters, [70](#)
- IDP_SIGNATURE
 - Run-time Parameters, [69](#)
- IDP_SMOOTHING
 - Pre-initialization Parameters, [64](#)
- IDP_VIEW_CLR
 - Run-time Parameters, [70](#)
- IDP_VIEW_CURSOR_X
 - Run-time Parameters, [70](#)
- IDP_VIEW_CURSOR_Y
 - Run-time Parameters, [70](#)
- IDP_VIEW_FIT
 - Run-time Parameters, [69](#)
- IDP_VIEW_SCALE
 - Run-time Parameters, [69](#)
- IDP_VIEW_X
 - Run-time Parameters, [69](#)
- IDP_VIEW_Y
 - Run-time Parameters, [69](#)
- IDPC_CMD_CORR_CALC
 - lpxDisplay Command Parameters, [85](#)
- IDPC_CMD_DUMP_OFF
 - lpxDisplay Command Parameters, [87](#)
- IDPC_CMD_DUMP_ON
 - lpxDisplay Command Parameters, [87](#)
- IDPC_CMD_FILTER_ADD
 - lpxDisplay Command Parameters, [87](#)
- IDPC_CMD_FILTER_DEL
 - lpxDisplay Command Parameters, [87](#)
- IDPC_CMD_MANAGED_OFF
 - lpxDisplay Command Parameters, [86](#)
- IDPC_CMD_MANAGED_ON
 - lpxDisplay Command Parameters, [86](#)
- IDPC_CMD_MENU_SHOW
 - lpxDisplay Command Parameters, [88](#)
- IDPC_CMD_OVERLAY_HIDE
 - lpxDisplay Command Parameters, [86](#)
- IDPC_CMD_OVERLAY_SHOW
 - lpxDisplay Command Parameters, [86](#)
- IDPC_CMD_PROC_ADD
 - lpxDisplay Command Parameters, [88](#)
- IDPC_CMD_PROC_DEL
 - lpxDisplay Command Parameters, [88](#)
- IDPC_CMD_VIEW_ATCENTER
 - lpxDisplay Command Parameters, [85](#)
- IDPC_CMD_VIEW_AT
 - lpxDisplay Command Parameters, [85](#)
- IDPC_CMD_VIEW_PARAMS
 - lpxDisplay Command Parameters, [85](#)
- IDPC_CMD_VIEW_ZOOM_IN
 - lpxDisplay Command Parameters, [84](#)
- IDPC_CMD_VIEW_ZOOM_OUT
 - lpxDisplay Command Parameters, [84](#)
- IDPC_SET_CORRECTION
 - lpxDisplay Command Parameters, [84](#)
- II_PIXEL_ALIGNMENT
 - lpxPixelFormat Header, [34](#)
- II_PIXEL_BITS
 - lpxPixelFormat Header, [36](#)
- II_PIXEL_CHROMATICITY
 - lpxPixelFormat Header, [35](#)
- II_PIXEL_TYPE_DEFINES
 - lpxPixelFormat Header, [37](#)
- IIConvert
 - lpxImageConverter, [188](#)
- IPX_USER_DATA
 - lpxUserData Header, [52](#)
- IPXD_CCLR_CHANGED
 - Notifications, [90](#)
- IPXD_CURSOR_MOVED
 - Notifications, [90](#)
- IPXD_ERROR_OPENGL
 - Notifications, [90](#)
- IPXD_FIT_FILL
 - Fit Modes and Mouse Processing, [92](#)
- IPXD_FIT_FULLSIZE
 - Fit Modes and Mouse Processing, [93](#)
- IPXD_FIT_NONE
 - Fit Modes and Mouse Processing, [92](#)
- IPXD_FIT_WINDOW
 - Fit Modes and Mouse Processing, [92](#)
- IPXD_KEY_DOWN
 - Notifications, [90](#)
- IPXD_LBUTTON_DOWN
 - Notifications, [89](#)
- IPXD_LBUTTON_UP
 - Notifications, [89](#)
- IPXD_MOUSE_DEFAULT
 - Fit Modes and Mouse Processing, [93](#)
- IPXD_MOUSE_LOCK
 - Fit Modes and Mouse Processing, [93](#)
- IPXD_MOUSE_SKIP
 - Fit Modes and Mouse Processing, [93](#)
- IPXD_PLAYBACK_FAILED
 - Notifications, [90](#)
- IPXD_RBUTTON_DOWN
 - Notifications, [90](#)
- IPXD_VIEW_CHANGED
 - Notifications, [90](#)
- ISP_ADD_PALETTE

- IpxSerializer Parameters, [109](#)
- ISP_JPEG_QUALITY
 - IpxSerializer Parameters, [107](#)
- ISP_MAX_QUANTIZER
 - IpxSerializer Parameters, [107](#)
- ISP_MIN_QUANTIZER
 - IpxSerializer Parameters, [107](#)
- ISP_MOVIE_COMPRESSORS
 - IpxSerializer Parameters, [108](#)
- ISP_MOVIE_COMPRESSOR
 - IpxSerializer Parameters, [108](#)
- ISP_NO_REALLOC
 - IpxSerializer Parameters, [107](#)
- ISP_TICKS_PER_SEC
 - IpxSerializer Parameters, [108](#)
- id
 - IpxUserData, [208](#)
- Image Converter Reference, [29](#)
- Image Unpacker Reference, [31](#)
- imageData
 - IpxImage, [186](#)
- imageDataOrigin
 - IpxImage, [186](#)
- imageID
 - IpxImage, [186](#)
- imageSize
 - IpxImage, [185](#)
- Imperx Demosaicing SDK Overview, [7](#)
- Initialize
 - IpxDisplay, [177](#)
- IpxAlloc
 - IpxImageApi Header, [15](#)
- IpxBayer, [161](#)
 - AllocData, [165](#)
 - ConvertImage, [163](#)
 - CreateComponent, [162](#)
 - DeleteComponent, [162](#)
 - GetComponent, [162](#)
 - ReleaseData, [165](#)
- IpxBayer C++ Class, [57](#)
- IpxBayer C-Interface Functions, [58](#)
 - IpxBayer_AllocData, [60](#)
 - IpxBayer_ConvertImage, [59](#)
 - IpxBayer_CreateComponent, [58](#)
 - IpxBayer_DeleteComponent, [59](#)
 - IpxBayer_GetComponent, [59](#)
 - IpxBayer_ReleaseData, [60](#)
- IpxBayer IpxComponent Header, [8](#)
- IpxBayer_AllocData
 - IpxBayer C-Interface Functions, [60](#)
- IpxBayer_ConvertImage
 - IpxBayer C-Interface Functions, [59](#)
- IpxBayer_CreateComponent
 - IpxBayer C-Interface Functions, [58](#)
- IpxBayer_DeleteComponent
 - IpxBayer C-Interface Functions, [59](#)
- IpxBayer_GetComponent
 - IpxBayer C-Interface Functions, [59](#)
- IpxBayer_ReleaseData
 - IpxBayer C-Interface Functions, [60](#)
- IpxCheckChannelNames
 - IpxPixelType Header, [44](#)
- IpxClonImage
 - IpxImageApi Header, [22](#)
- IpxClonImageExt
 - IpxImageApi Header, [23](#)
- IpxComponent, [166](#)
 - ~IpxComponent, [167](#)
 - GetComponentTypeID, [167](#)
 - GetParamArray, [173](#)
 - GetParamAsString, [168](#)
 - GetParamBool, [171](#)
 - GetParamCount, [167](#)
 - GetParamFloat, [172](#)
 - GetParamInt, [172](#)
 - GetParamName, [167](#)
 - GetParamString, [173](#)
 - RunCommand, [174](#)
 - SetParamArray, [171](#)
 - SetParamAsString, [168](#)
 - SetParamBool, [169](#)
 - SetParamFloat, [170](#)
 - SetParamInt, [169](#)
 - SetParamString, [170](#)
- IpxConvertChannelStr
 - IpxPixelType Header, [44](#)
- IpxCopyImage
 - IpxImageApi Header, [24](#)
- IpxCopyImageChannelChar
 - IpxImageApi Header, [25](#)
- IpxCopyImageChannelFloat
 - IpxImageApi Header, [27](#)
- IpxCopyImageChannelInt
 - IpxImageApi Header, [26](#)
- IpxCopyImageChannelShort
 - IpxImageApi Header, [26](#)
- IpxCopyImageHeader
 - IpxImageApi Header, [23](#)
- IpxCreateEmptyImageHeader
 - IpxImageApi Header, [17](#)
- IpxCreatImage
 - IpxImageApi Header, [20](#)
- IpxCreatImageData
 - IpxImageApi Header, [19](#)
- IpxCreatImageHeader
 - IpxImageApi Header, [17](#)
- IpxDisplay, [174](#)
 - ConvertImage, [180](#)

- CreateComponent, [175](#)
- DeleteComponent, [175](#)
- DisplayImage, [180](#)
- DisplayVideo, [179](#)
- GetComponent, [176](#)
- GetSystemInfo, [177](#)
- Initialize, [177](#)
- SetVideoMode, [179](#)
- Translate, [182](#)
- IpxDisplay C++ Class, [94](#)
- IpxDisplay C-Interface Functions, [95](#)
 - IpxDisplay_ConvertImage, [101](#)
 - IpxDisplay_CreateComponent, [95](#)
 - IpxDisplay_DeleteComponent, [96](#)
 - IpxDisplay_DisplayImage, [100](#)
 - IpxDisplay_DisplayVideo, [98](#)
 - IpxDisplay_GetComponent, [97](#)
 - IpxDisplay_Initialize, [97](#)
- IpxDisplay Command Parameters, [83](#)
 - IDPC_CMD_CORR_CALC, [85](#)
 - IDPC_CMD_DUMP_OFF, [87](#)
 - IDPC_CMD_DUMP_ON, [87](#)
 - IDPC_CMD_FILTER_ADD, [87](#)
 - IDPC_CMD_FILTER_DEL, [87](#)
 - IDPC_CMD_MANAGED_OFF, [86](#)
 - IDPC_CMD_MANAGED_ON, [86](#)
 - IDPC_CMD_MENU_SHOW, [88](#)
 - IDPC_CMD_OVERLAY_HIDE, [86](#)
 - IDPC_CMD_OVERLAY_SHOW, [86](#)
 - IDPC_CMD_PROC_ADD, [88](#)
 - IDPC_CMD_PROC_DEL, [88](#)
 - IDPC_CMD_VIEW_ATCENTER, [85](#)
 - IDPC_CMD_VIEW_AT, [85](#)
 - IDPC_CMD_VIEW_PARAMS, [85](#)
 - IDPC_CMD_VIEW_ZOOM_IN, [84](#)
 - IDPC_CMD_VIEW_ZOOM_OUT, [84](#)
 - IDPC_SET_CORRECTION, [84](#)
- IpxDisplay IpxComponent Header, [9](#)
- IpxDisplay_ConvertImage
 - IpxDisplay C-Interface Functions, [101](#)
- IpxDisplay_CreateComponent
 - IpxDisplay C-Interface Functions, [95](#)
- IpxDisplay_DeleteComponent
 - IpxDisplay C-Interface Functions, [96](#)
- IpxDisplay_DisplayImage
 - IpxDisplay C-Interface Functions, [100](#)
- IpxDisplay_DisplayVideo
 - IpxDisplay C-Interface Functions, [98](#)
- IpxDisplay_GetComponent
 - IpxDisplay C-Interface Functions, [97](#)
- IpxDisplay_Initialize
 - IpxDisplay C-Interface Functions, [97](#)
- IpxFree
 - IpxImageApi Header, [16](#)
- IpxGetChannelIndex
 - IpxPixelFormat Header, [43](#)
- IpxGetChannelName
 - IpxPixelFormat Header, [45](#)
- IpxGetChannelSequence
 - IpxPixelFormat Header, [41](#)
- IpxGetChannelsDepth
 - IpxPixelFormat Header, [42](#)
- IpxGetChannelsNumber
 - IpxPixelFormat Header, [42](#)
- IpxGetColorModelDescr
 - IpxPixelFormat Header, [40](#)
- IpxGetColorModelDescription
 - IpxPixelFormat Header, [40](#)
- IpxGetColorModelName
 - IpxPixelFormat Header, [41](#)
- IpxGetPixelFormat
 - IpxPixelFormat Header, [41](#)
- IpxGetPixelTypesNumber
 - IpxPixelFormat Header, [39](#)
- IpxGetRowSize
 - IpxPixelFormat Header, [38](#)
- IpxGetRowSizeUnaligned
 - IpxPixelFormat Header, [38](#)
- IpxGetStartPosition
 - IpxPixelFormat Header, [43](#)
- IpxImage, [183](#)
 - height, [185](#)
 - imageData, [186](#)
 - imageDataOrigin, [186](#)
 - imageID, [186](#)
 - imageSize, [185](#)
 - nSize, [184](#)
 - origin, [185](#)
 - pixelTypeDescr, [185](#)
 - rowSize, [185](#)
 - timestamp, [185](#)
 - userData, [186](#)
 - version, [184](#)
 - width, [185](#)
- IpxImage Header, [11](#)
 - IpxInitPixelFormatDescr, [11](#)
- IpxImageApi Header, [13](#)
 - IpxAlloc, [15](#)
 - IpxCloneImage, [22](#)
 - IpxCloneImageExt, [23](#)
 - IpxCopyImage, [24](#)
 - IpxCopyImageChannelChar, [25](#)
 - IpxCopyImageChannelFloat, [27](#)
 - IpxCopyImageChannelInt, [26](#)
 - IpxCopyImageChannelShort, [26](#)
 - IpxCopyImageHeader, [23](#)
 - IpxCreateEmptyImageHeader, [17](#)
 - IpxCreateImage, [20](#)

- lpxCreateImageData, 19
- lpxCreateImageHeader, 17
- lpxFree, 16
- lpxInitImageHeader, 18
- lpxReleaseImage, 21
- lpxReleaseImageHeader, 21
- lpxSetMemoryManager, 15
- PAllocFunc, 14
- PFreeFunc, 14
- lpxImageConverter, 186
 - ConvertImage, 188
 - CreateComponent, 187
 - DeleteComponent, 187
 - GetComponent, 188
 - IIConvert, 188
- lpxImageConverter C-Interface Functions, 102
 - lpxImageConverter_ConvertImage, 103
 - lpxImageConverter_CreateComponent, 102
 - lpxImageConverter_DeleteComponent, 103
 - lpxImageConverter_GetComponent, 103
 - lpxImageConverter_IIConvert, 104
- lpxImageConverter_ConvertImage
 - lpxImageConverter C-Interface Functions, 103
- lpxImageConverter_CreateComponent
 - lpxImageConverter C-Interface Functions, 102
- lpxImageConverter_DeleteComponent
 - lpxImageConverter C-Interface Functions, 103
- lpxImageConverter_GetComponent
 - lpxImageConverter C-Interface Functions, 103
- lpxImageConverter_IIConvert
 - lpxImageConverter C-Interface Functions, 104
- lpxImageSerializer, 189
 - CreateComponent, 190
 - DeleteComponent, 190
 - FinishRecord, 193
 - Free, 195
 - GetComponent, 191
 - GetImageHeader, 194
 - Load, 194
 - Save, 193
 - StartMovieRecord, 192
 - StartSeriesRecord, 192
- lpxImageSerializer C++ Class, 110
- lpxImageSerializer C-Interface Functions, 111
 - lpxImageSerializer_CreateComponent, 112
 - lpxImageSerializer_DeleteComponent, 112
 - lpxImageSerializer_FinishRecord, 114
 - lpxImageSerializer_GetComponent, 112
 - lpxImageSerializer_Load, 115
 - lpxImageSerializer_Save, 114
 - lpxImageSerializer_StartMovieRecord, 113
 - lpxImageSerializer_StartSeriesRecord, 113
- lpxImageSerializer lpxComponent Header, 30
- lpxImageSerializer_CreateComponent
 - lpxImageSerializer C-Interface Functions, 112
- lpxImageSerializer_DeleteComponent
 - lpxImageSerializer C-Interface Functions, 112
- lpxImageSerializer_FinishRecord
 - lpxImageSerializer C-Interface Functions, 114
- lpxImageSerializer_GetComponent
 - lpxImageSerializer C-Interface Functions, 112
- lpxImageSerializer_Load
 - lpxImageSerializer C-Interface Functions, 115
- lpxImageSerializer_Save
 - lpxImageSerializer C-Interface Functions, 114
- lpxImageSerializer_StartMovieRecord
 - lpxImageSerializer C-Interface Functions, 113
- lpxImageSerializer_StartSeriesRecord
 - lpxImageSerializer C-Interface Functions, 113
- lpxImageUnpacker, 195
 - CreateComponent, 196
 - DeleteComponent, 196
 - GetComponent, 197
 - Unpack, 197
- lpxImageUnpacker C-Interface Functions, 116
 - lpxImageUnpacker_CreateComponent, 116
 - lpxImageUnpacker_DeleteComponent, 116
 - lpxImageUnpacker_GetComponent, 117
 - lpxImageUnpacker_Unpack, 117
- lpxImageUnpacker_CreateComponent
 - lpxImageUnpacker C-Interface Functions, 116
- lpxImageUnpacker_DeleteComponent
 - lpxImageUnpacker C-Interface Functions, 116
- lpxImageUnpacker_GetComponent
 - lpxImageUnpacker C-Interface Functions, 117
- lpxImageUnpacker_Unpack
 - lpxImageUnpacker C-Interface Functions, 117
- lpxImgProcessor, 198
- lpxInitImageHeader
 - lpxImageApi Header, 18
- lpxInitPixelFormatDescr
 - lpxImage Header, 11
- lpxIsGroup
 - lpxPixelFormat Header, 39
- lpxIsPixelFormat
 - lpxPixelFormat Header, 39
- lpxPixelFormat Header, 32
 - II_PIXEL_ALIGNMENT, 34
 - II_PIXEL_BITS, 36
 - II_PIXEL_CHROMATICITY, 35
 - II_PIXEL_TYPE_DEFINES, 37
 - lpxCheckChannelNames, 44
 - lpxConvertChannelStr, 44
 - lpxGetChannelIndex, 43
 - lpxGetChannelName, 45
 - lpxGetChannelSequence, 41
 - lpxGetChannelsDepth, 42
 - lpxGetChannelsNumber, 42

- [IpxGetColorModelDescr](#), [40](#)
 - [IpxGetColorModelDescription](#), [40](#)
 - [IpxGetColorModelName](#), [41](#)
 - [IpxGetPixelFormat](#), [41](#)
 - [IpxGetPixelTypesNumber](#), [39](#)
 - [IpxGetRowSize](#), [38](#)
 - [IpxGetRowSizeUnaligned](#), [38](#)
 - [IpxGetStartPosition](#), [43](#)
 - [IpxIsGroup](#), [39](#)
 - [IpxIsPixelFormat](#), [39](#)
- [IpxPixelFormatDescr](#), [198](#)
 - [channels](#), [199](#)
 - [depth](#), [199](#)
 - [pixAlign](#), [199](#)
 - [pixSigned](#), [199](#)
 - [pixSize](#), [200](#)
 - [pixelType](#), [199](#)
- [IpxPoint](#), [200](#)
 - [x](#), [200](#)
 - [y](#), [200](#)
- [IpxRect](#), [201](#)
 - [height](#), [202](#)
 - [width](#), [201](#)
 - [x](#), [201](#)
 - [y](#), [201](#)
- [IpxReleaseImage](#)
 - [IpxImageApi Header](#), [21](#)
- [IpxReleaseImageHeader](#)
 - [IpxImageApi Header](#), [21](#)
- [IpxSerializer Parameters](#), [106](#)
 - [ISP_ADD_PALETTE](#), [109](#)
 - [ISP_JPEG_QUALITY](#), [107](#)
 - [ISP_MAX_QUANTIZER](#), [107](#)
 - [ISP_MIN_QUANTIZER](#), [107](#)
 - [ISP_MOVIE_COMPRESSORS](#), [108](#)
 - [ISP_MOVIE_COMPRESSOR](#), [108](#)
 - [ISP_NO_REALLOC](#), [107](#)
 - [ISP_TICKS_PER_SEC](#), [108](#)
- [IpxSetMemoryManager](#)
 - [IpxImageApi Header](#), [15](#)
- [IpxSize](#), [202](#)
 - [height](#), [202](#)
 - [width](#), [202](#)
- [IpxToolBase Header](#), [46](#)
- [IpxTrueSense](#), [203](#)
 - [AllocData](#), [207](#)
 - [ConvertImage](#), [205](#)
 - [CreateComponent](#), [204](#)
 - [DeleteComponent](#), [204](#)
 - [GetComponent](#), [204](#)
 - [ReleaseData](#), [207](#)
- [IpxTrueSense C++ Class](#), [156](#)
- [IpxTrueSense C-Interface Functions](#), [157](#)
 - [IpxTrueSense_AllocData](#), [159](#)
 - [IpxTrueSense_ConvertImage](#), [159](#)
 - [IpxTrueSense_CreateComponent](#), [157](#)
 - [IpxTrueSense_DeleteComponent](#), [158](#)
 - [IpxTrueSense_GetComponent](#), [158](#)
 - [IpxTrueSense_ReleaseData](#), [160](#)
- [IpxTrueSense IpxComponent Header](#), [50](#)
- [IpxTrueSense_AllocData](#)
 - [IpxTrueSense C-Interface Functions](#), [159](#)
- [IpxTrueSense_ConvertImage](#)
 - [IpxTrueSense C-Interface Functions](#), [159](#)
- [IpxTrueSense_CreateComponent](#)
 - [IpxTrueSense C-Interface Functions](#), [157](#)
- [IpxTrueSense_DeleteComponent](#)
 - [IpxTrueSense C-Interface Functions](#), [158](#)
- [IpxTrueSense_GetComponent](#)
 - [IpxTrueSense C-Interface Functions](#), [158](#)
- [IpxTrueSense_ReleaseData](#)
 - [IpxTrueSense C-Interface Functions](#), [160](#)
- [IpxUserData](#), [208](#)
 - [createdIpx](#), [209](#)
 - [data](#), [209](#)
 - [id](#), [208](#)
 - [pNext](#), [209](#)
 - [size](#), [208](#)
 - [type](#), [208](#)
- [IpxUserData Header](#), [52](#)
 - [IPX_USER_DATA](#), [52](#)
- [Load](#)
 - [IpxImageSerializer](#), [194](#)
- [nSize](#)
 - [IpxImage](#), [184](#)
- [Notifications](#), [89](#)
 - [IPXD_CCLR_CHANGED](#), [90](#)
 - [IPXD_CURSOR_MOVED](#), [90](#)
 - [IPXD_ERROR_OPENGL](#), [90](#)
 - [IPXD_KEY_DOWN](#), [90](#)
 - [IPXD_LBUTTON_DOWN](#), [89](#)
 - [IPXD_LBUTTON_UP](#), [89](#)
 - [IPXD_PLAYBACK_FAILED](#), [90](#)
 - [IPXD_RBUTTON_DOWN](#), [90](#)
 - [IPXD_VIEW_CHANGED](#), [90](#)
- [origin](#)
 - [IpxImage](#), [185](#)
- [Overlay Text Parameters](#), [78](#)
 - [IDP_OVERLAY_BGMODE](#), [79](#)
 - [IDP_OVERLAY_COLOR](#), [79](#)
 - [IDP_OVERLAY_FONT](#), [79](#)
 - [IDP_OVERLAY_INDEX](#), [78](#)
 - [IDP_OVERLAY_POS](#), [79](#)
 - [IDP_OVERLAY_TEXT](#), [80](#)
- [PAllocFunc](#)

- lpxImageApi Header, [14](#)
- PFreeFunc
 - lpxImageApi Header, [14](#)
- pNext
 - lpxUserData, [209](#)
- pixAlign
 - lpxPixelTypeDescr, [199](#)
- pixSigned
 - lpxPixelTypeDescr, [199](#)
- pixSize
 - lpxPixelTypeDescr, [200](#)
- pixelType
 - lpxPixelTypeDescr, [199](#)
- pixelTypeDescr
 - lpxImage, [185](#)
- Pre-initialization Parameters, [62](#)
 - IDP_BACKGROUND, [63](#)
 - IDP_COMMAND_WINDOW, [66](#)
 - IDP_GDI_BAYER, [65](#)
 - IDP_GDI_TRUESENSE, [65](#)
 - IDP_INIT_AT_X, [63](#)
 - IDP_INIT_AT_Y, [64](#)
 - IDP_INIT_FIT, [63](#)
 - IDP_OGL_BAYER, [64](#)
 - IDP_OGL_TRUESENSE, [65](#)
 - IDP_OVERLAY_FONT_DESC_0, [66](#)
 - IDP_OVERLAY_FONT_DESC_1, [66](#)
 - IDP_OVERLAY_FONT_DESC_2, [67](#)
 - IDP_OVERLAY_FONT_DESC_3, [67](#)
 - IDP_SMOOTHING, [64](#)
- ReleaseData
 - lpxBayer, [165](#)
 - lpxTrueSense, [207](#)
- rowSize
 - lpxImage, [185](#)
- Run-time Parameters, [68](#)
 - IDP_MANAGED_FPS, [69](#)
 - IDP_MANAGED_STATE, [70](#)
 - IDP_MENU_CMD, [71](#)
 - IDP_MENU_X, [71](#)
 - IDP_MENU_Y, [71](#)
 - IDP_PROC_PROCESSOR_TYPE, [70](#)
 - IDP_PROC_PROCESSOR, [70](#)
 - IDP_SIGNATURE, [69](#)
 - IDP_VIEW_CLR, [70](#)
 - IDP_VIEW_CURSOR_X, [70](#)
 - IDP_VIEW_CURSOR_Y, [70](#)
 - IDP_VIEW_FIT, [69](#)
 - IDP_VIEW_SCALE, [69](#)
 - IDP_VIEW_X, [69](#)
 - IDP_VIEW_Y, [69](#)
- RunCommand
 - lpxComponent, [174](#)
- Save
 - lpxImageSerializer, [193](#)
- SetParamArray
 - lpxComponent, [171](#)
- SetParamAsString
 - lpxComponent, [168](#)
- SetParamBool
 - lpxComponent, [169](#)
- SetParamFloat
 - lpxComponent, [170](#)
- SetParamInt
 - lpxComponent, [169](#)
- SetParamString
 - lpxComponent, [170](#)
- SetVideoMode
 - lpxDisplay, [179](#)
- size
 - lpxUserData, [208](#)
- Software Image Correction Parameters, [72](#)
 - IDP_CORR_GAIN_B, [73](#)
 - IDP_CORR_GAIN_G, [73](#)
 - IDP_CORR_GAIN_R, [73](#)
 - IDP_CORR_GAMMA, [75](#)
 - IDP_CORR_MODE, [73](#)
 - IDP_CORR_OFFSETS_B, [74](#)
 - IDP_CORR_OFFSETS_G, [74](#)
 - IDP_CORR_OFFSETS_R, [74](#)
- StartMovieRecord
 - lpxImageSerializer, [192](#)
- StartSeriesRecord
 - lpxImageSerializer, [192](#)
- TRUES_OPENGL_MHC
 - TS CFA Demosaicing algorithm Parameters, [121](#)
- TRUES_OPENGL_MMA
 - TS CFA Demosaicing algorithm Parameters, [121](#)
- TS CFA Demosaicing algorithm Parameters, [119](#)
 - TRUES_OPENGL_MHC, [121](#)
 - TRUES_OPENGL_MMA, [121](#)
 - TS_ALGO_NUM, [120](#)
 - TS_ALGO_TYPE, [119](#)
 - TS_NOREALLOC, [120](#)
 - TSABAYERLIKE, [120](#)
 - TSAMEDIUM, [121](#)
 - TSQUALITY, [121](#)
 - TSASIMPLEF, [120](#)
 - TSASIMPLES, [120](#)
- TS Coefficients Parameters, [148](#)
 - TS_BB_COEFF, [151](#)
 - TS_BG_COEFF, [151](#)
 - TS_BR_COEFF, [150](#)
 - TS_GB_COEFF, [150](#)
 - TS_GG_COEFF, [150](#)
 - TS_GR_COEFF, [149](#)

- TS_RB_COEFF, [149](#)
- TS_RG_COEFF, [149](#)
- TS_RR_COEFF, [149](#)
- TS Gain Parameters, [126](#)
 - TS_ANALOG_GAIN, [129](#)
 - TS_BLUE_GAIN, [128](#)
 - TS_GLOBAL_GAIN, [128](#)
 - TS_GREEN_GAIN, [127](#)
 - TS_ISO_ANALOGGAIN_0, [129](#)
 - TS_ISO_ANALOGGAIN_1, [129](#)
 - TS_ISO_ANALOGGAIN_2, [130](#)
 - TS_ISO_ANALOGGAIN_3, [130](#)
 - TS_ISO_ANALOGGAIN_4, [130](#)
 - TS_PAN_GAIN, [128](#)
 - TS_RED_GAIN, [127](#)
- TS ISO Color Intercept Parameters, [139](#)
 - TS_ISO_COLORINTERCEPT_0, [140](#)
 - TS_ISO_COLORINTERCEPT_1, [140](#)
 - TS_ISO_COLORINTERCEPT_2, [140](#)
 - TS_ISO_COLORINTERCEPT_3, [140](#)
 - TS_ISO_COLORINTERCEPT_4, [141](#)
- TS ISO Color Slope Parameters, [136](#)
 - TS_ISO_COLORSCOPE_0, [137](#)
 - TS_ISO_COLORSCOPE_1, [137](#)
 - TS_ISO_COLORSCOPE_2, [137](#)
 - TS_ISO_COLORSCOPE_3, [137](#)
 - TS_ISO_COLORSCOPE_4, [138](#)
- TS ISO Panchromatic Channel Parameters, [131](#)
 - TS_ISO_PANINTERCEPT_0, [133](#)
 - TS_ISO_PANINTERCEPT_1, [134](#)
 - TS_ISO_PANINTERCEPT_2, [134](#)
 - TS_ISO_PANINTERCEPT_3, [134](#)
 - TS_ISO_PANINTERCEPT_4, [135](#)
 - TS_ISO_PANSLOPE_0, [132](#)
 - TS_ISO_PANSLOPE_1, [132](#)
 - TS_ISO_PANSLOPE_2, [132](#)
 - TS_ISO_PANSLOPE_3, [133](#)
 - TS_ISO_PANSLOPE_4, [133](#)
- TS Misc Parameters, [122](#)
 - TS_DARKFLOOR, [125](#)
 - TS_HORIZ_MIRRORED, [123](#)
 - TS_IMP_FILTER_ENABLED, [124](#)
 - TS_MONO_ENABLED, [124](#)
 - TS_NORM_EN, [123](#)
 - TS_SHARPNESS_ENABLED, [124](#)
 - TS_THREADS_NUM, [123](#)
 - TS_VER_MIRRORED, [123](#)
- TS Noise Threshold Parameters, [154](#)
 - TS_HIGH_LUMA_NOISE, [154](#)
 - TS_LOW_LUMA_NOISE, [154](#)
- TS Sharpen Parameters, [152](#)
 - TS_MAX_SHARPEN, [152](#)
 - TS_SHARPEN_PARAM, [152](#)
- TS Sigma Filter Parameters, [142](#)
 - TS_COLOR_RADIUS0, [145](#)
 - TS_COLOR_RADIUS1, [145](#)
 - TS_COLOR_RADIUS2, [146](#)
 - TS_COLOR_SIGMA0, [146](#)
 - TS_COLOR_SIGMA1, [146](#)
 - TS_COLOR_SIGMA2, [147](#)
 - TS_PAN_RADIUS0, [143](#)
 - TS_PAN_RADIUS1, [143](#)
 - TS_PAN_RADIUS2, [144](#)
 - TS_PAN_SIGMA0, [144](#)
 - TS_PAN_SIGMA1, [144](#)
 - TS_PAN_SIGMA2, [145](#)
- TS_ALGO_NUM
 - TS CFA Demosaicing algorithm Parameters, [120](#)
- TS_ALGO_TYPE
 - TS CFA Demosaicing algorithm Parameters, [119](#)
- TS_ANALOG_GAIN
 - TS Gain Parameters, [129](#)
- TS_BB_COEFF
 - TS Coefficients Parameters, [151](#)
- TS_BG_COEFF
 - TS Coefficients Parameters, [151](#)
- TS_BLUE_GAIN
 - TS Gain Parameters, [128](#)
- TS_BR_COEFF
 - TS Coefficients Parameters, [150](#)
- TS_COLOR_RADIUS0
 - TS Sigma Filter Parameters, [145](#)
- TS_COLOR_RADIUS1
 - TS Sigma Filter Parameters, [145](#)
- TS_COLOR_RADIUS2
 - TS Sigma Filter Parameters, [146](#)
- TS_COLOR_SIGMA0
 - TS Sigma Filter Parameters, [146](#)
- TS_COLOR_SIGMA1
 - TS Sigma Filter Parameters, [146](#)
- TS_COLOR_SIGMA2
 - TS Sigma Filter Parameters, [147](#)
- TS_DARKFLOOR
 - TS Misc Parameters, [125](#)
- TS_GB_COEFF
 - TS Coefficients Parameters, [150](#)
- TS_GG_COEFF
 - TS Coefficients Parameters, [150](#)
- TS_GLOBAL_GAIN
 - TS Gain Parameters, [128](#)
- TS_GR_COEFF
 - TS Coefficients Parameters, [149](#)
- TS_GREEN_GAIN
 - TS Gain Parameters, [127](#)
- TS_HIGH_LUMA_NOISE
 - TS Noise Threshold Parameters, [154](#)
- TS_HORIZ_MIRRORED
 - TS Misc Parameters, [123](#)

- TS_IMP_FILTER_ENABLED
 - TS Misc Parameters, [124](#)
- TS_ISO_ANALOGGAIN_0
 - TS Gain Parameters, [129](#)
- TS_ISO_ANALOGGAIN_1
 - TS Gain Parameters, [129](#)
- TS_ISO_ANALOGGAIN_2
 - TS Gain Parameters, [130](#)
- TS_ISO_ANALOGGAIN_3
 - TS Gain Parameters, [130](#)
- TS_ISO_ANALOGGAIN_4
 - TS Gain Parameters, [130](#)
- TS_ISO_COLORINTERCEPT_0
 - TS ISO Color Intercept Parameters, [140](#)
- TS_ISO_COLORINTERCEPT_1
 - TS ISO Color Intercept Parameters, [140](#)
- TS_ISO_COLORINTERCEPT_2
 - TS ISO Color Intercept Parameters, [140](#)
- TS_ISO_COLORINTERCEPT_3
 - TS ISO Color Intercept Parameters, [140](#)
- TS_ISO_COLORINTERCEPT_4
 - TS ISO Color Intercept Parameters, [141](#)
- TS_ISO_COLORSLOPE_0
 - TS ISO Color Slope Parameters, [137](#)
- TS_ISO_COLORSLOPE_1
 - TS ISO Color Slope Parameters, [137](#)
- TS_ISO_COLORSLOPE_2
 - TS ISO Color Slope Parameters, [137](#)
- TS_ISO_COLORSLOPE_3
 - TS ISO Color Slope Parameters, [137](#)
- TS_ISO_COLORSLOPE_4
 - TS ISO Color Slope Parameters, [138](#)
- TS_ISO_PANINTERCEPT_0
 - TS ISO Panchromatic Channel Parameters, [133](#)
- TS_ISO_PANINTERCEPT_1
 - TS ISO Panchromatic Channel Parameters, [134](#)
- TS_ISO_PANINTERCEPT_2
 - TS ISO Panchromatic Channel Parameters, [134](#)
- TS_ISO_PANINTERCEPT_3
 - TS ISO Panchromatic Channel Parameters, [134](#)
- TS_ISO_PANINTERCEPT_4
 - TS ISO Panchromatic Channel Parameters, [135](#)
- TS_ISO_PANSLOPE_0
 - TS ISO Panchromatic Channel Parameters, [132](#)
- TS_ISO_PANSLOPE_1
 - TS ISO Panchromatic Channel Parameters, [132](#)
- TS_ISO_PANSLOPE_2
 - TS ISO Panchromatic Channel Parameters, [132](#)
- TS_ISO_PANSLOPE_3
 - TS ISO Panchromatic Channel Parameters, [133](#)
- TS_ISO_PANSLOPE_4
 - TS ISO Panchromatic Channel Parameters, [133](#)
- TS_LOW_LUMA_NOISE
 - TS Noise Threshold Parameters, [154](#)
- TS_MAX_SHARPEN
 - TS Sharpen Parameters, [152](#)
- TS_MONO_ENABLED
 - TS Misc Parameters, [124](#)
- TS_NOREALLOC
 - TS CFA Demosaicing algorithm Parameters, [120](#)
- TS_NORM_EN
 - TS Misc Parameters, [123](#)
- TS_PAN_GAIN
 - TS Gain Parameters, [128](#)
- TS_PAN_RADIUS0
 - TS Sigma Filter Parameters, [143](#)
- TS_PAN_RADIUS1
 - TS Sigma Filter Parameters, [143](#)
- TS_PAN_RADIUS2
 - TS Sigma Filter Parameters, [144](#)
- TS_PAN_SIGMA0
 - TS Sigma Filter Parameters, [144](#)
- TS_PAN_SIGMA1
 - TS Sigma Filter Parameters, [144](#)
- TS_PAN_SIGMA2
 - TS Sigma Filter Parameters, [145](#)
- TS_RB_COEFF
 - TS Coefficients Parameters, [149](#)
- TS_RED_GAIN
 - TS Gain Parameters, [127](#)
- TS_RG_COEFF
 - TS Coefficients Parameters, [149](#)
- TS_RR_COEFF
 - TS Coefficients Parameters, [149](#)
- TS_SHARPEN_PARAM
 - TS Sharpen Parameters, [152](#)
- TS_SHARPNESS_ENABLED
 - TS Misc Parameters, [124](#)
- TS_THREADS_NUM
 - TS Misc Parameters, [123](#)
- TS_VER_MIRRORED
 - TS Misc Parameters, [123](#)
- TSABAYERLIKE
 - TS CFA Demosaicing algorithm Parameters, [120](#)
- TSAMEDIUM
 - TS CFA Demosaicing algorithm Parameters, [121](#)
- TSAQUALITY
 - TS CFA Demosaicing algorithm Parameters, [121](#)
- TSASIMPLEF
 - TS CFA Demosaicing algorithm Parameters, [120](#)
- TSASIMPLES
 - TS CFA Demosaicing algorithm Parameters, [120](#)
- timestamp
 - lpxImage, [185](#)
- Translate
 - lpxDisplay, [182](#)
- Translate Flags, [91](#)
 - IDFL_IMG_SCR, [91](#)

- IDFL_SCR_IMG, [91](#)
- type
 - lpxUserData, [208](#)
- Unpack
 - lpxImageUnpacker, [197](#)
- userData
 - lpxImage, [186](#)
- version
 - lpxImage, [184](#)
- White Balance Correction Parameters, [76](#)
 - IDP_CALC_COEF_B, [77](#)
 - IDP_CALC_COEF_G, [77](#)
 - IDP_CALC_COEF_R, [76](#)
- width
 - lpxImage, [185](#)
 - lpxRect, [201](#)
 - lpxSize, [202](#)
- x
 - lpxPoint, [200](#)
 - lpxRect, [201](#)
- y
 - lpxPoint, [200](#)
 - lpxRect, [201](#)