



man pages section 3: Multimedia Library Functions

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 817-7519-10
January 2005

Copyright 2005 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, docs.sun.com, AnswerBook, AnswerBook2, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

U.S. Government Rights – Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 2005 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées du système Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux États-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, docs.sun.com, AnswerBook, AnswerBook2, et Solaris sont des marques de fabrique ou des marques déposées, de Sun Microsystems, Inc. aux États-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux États-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REpondre A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.



050203@10536



Contents

Preface 33

Multimedia Library Functions Part One 39

<code>mllib_free(3MLIB)</code>	40
<code>mllib_GraphicsBoundaryFill_8(3MLIB)</code>	41
<code>mllib_GraphicsDrawArc_8(3MLIB)</code>	42
<code>mllib_GraphicsDrawArc_A_8(3MLIB)</code>	43
<code>mllib_GraphicsDrawArc_X_8(3MLIB)</code>	44
<code>mllib_GraphicsDrawCircle_8(3MLIB)</code>	45
<code>mllib_GraphicsDrawCircle_A_8(3MLIB)</code>	46
<code>mllib_GraphicsDrawCircle_X_8(3MLIB)</code>	47
<code>mllib_GraphicsDrawEllipse_8(3MLIB)</code>	48
<code>mllib_GraphicsDrawEllipse_A_8(3MLIB)</code>	49
<code>mllib_GraphicsDrawEllipse_X_8(3MLIB)</code>	50
<code>mllib_GraphicsDrawLine_8(3MLIB)</code>	51
<code>mllib_GraphicsDrawLine_A_8(3MLIB)</code>	52
<code>mllib_GraphicsDrawLine_AG_8(3MLIB)</code>	53
<code>mllib_GraphicsDrawLine_AGZ_8(3MLIB)</code>	54
<code>mllib_GraphicsDrawLine_AZ_8(3MLIB)</code>	56
<code>mllib_GraphicsDrawLineFanSet_8(3MLIB)</code>	57
<code>mllib_GraphicsDrawLineFanSet_A_8(3MLIB)</code>	58
<code>mllib_GraphicsDrawLineFanSet_AG_8(3MLIB)</code>	59
<code>mllib_GraphicsDrawLineFanSet_AGZ_8(3MLIB)</code>	60
<code>mllib_GraphicsDrawLineFanSet_AZ_8(3MLIB)</code>	61
<code>mllib_GraphicsDrawLineFanSet_G_8(3MLIB)</code>	62
<code>mllib_GraphicsDrawLineFanSet_GZ_8(3MLIB)</code>	63

mllib_GraphicsDrawLineFanSet_X_8(3MLIB)	64
mllib_GraphicsDrawLineFanSet_Z_8(3MLIB)	65
mllib_GraphicsDrawLine_G_8(3MLIB)	66
mllib_GraphicsDrawLine_GZ_8(3MLIB)	67
mllib_GraphicsDrawLineSet_8(3MLIB)	69
mllib_GraphicsDrawLineSet_A_8(3MLIB)	70
mllib_GraphicsDrawLineSet_AG_8(3MLIB)	71
mllib_GraphicsDrawLineSet_AGZ_8(3MLIB)	72
mllib_GraphicsDrawLineSet_AZ_8(3MLIB)	73
mllib_GraphicsDrawLineSet_G_8(3MLIB)	74
mllib_GraphicsDrawLineSet_GZ_8(3MLIB)	75
mllib_GraphicsDrawLineSet_X_8(3MLIB)	76
mllib_GraphicsDrawLineSet_Z_8(3MLIB)	77
mllib_GraphicsDrawLineStripSet_8(3MLIB)	78
mllib_GraphicsDrawLineStripSet_A_8(3MLIB)	79
mllib_GraphicsDrawLineStripSet_AG_8(3MLIB)	80
mllib_GraphicsDrawLineStripSet_AGZ_8(3MLIB)	81
mllib_GraphicsDrawLineStripSet_AZ_8(3MLIB)	82
mllib_GraphicsDrawLineStripSet_G_8(3MLIB)	83
mllib_GraphicsDrawLineStripSet_GZ_8(3MLIB)	84
mllib_GraphicsDrawLineStripSet_X_8(3MLIB)	85
mllib_GraphicsDrawLineStripSet_Z_8(3MLIB)	86
mllib_GraphicsDrawLine_X_8(3MLIB)	87
mllib_GraphicsDrawLine_Z_8(3MLIB)	88
mllib_GraphicsDrawPoint_8(3MLIB)	89
mllib_GraphicsDrawPointSet_8(3MLIB)	90
mllib_GraphicsDrawPointSet_X_8(3MLIB)	91
mllib_GraphicsDrawPoint_X_8(3MLIB)	92
mllib_GraphicsDrawPolygon_8(3MLIB)	93
mllib_GraphicsDrawPolygon_A_8(3MLIB)	94
mllib_GraphicsDrawPolygon_AG_8(3MLIB)	95
mllib_GraphicsDrawPolygon_AGZ_8(3MLIB)	96
mllib_GraphicsDrawPolygon_AZ_8(3MLIB)	97
mllib_GraphicsDrawPolygon_G_8(3MLIB)	98
mllib_GraphicsDrawPolygon_GZ_8(3MLIB)	99
mllib_GraphicsDrawPolygon_X_8(3MLIB)	100
mllib_GraphicsDrawPolygon_Z_8(3MLIB)	101
mllib_GraphicsDrawPolyline_8(3MLIB)	102

mllib_GraphicsDrawPolyline_A_8(3MLIB)	103
mllib_GraphicsDrawPolyline_AG_8(3MLIB)	104
mllib_GraphicsDrawPolyline_AGZ_8(3MLIB)	105
mllib_GraphicsDrawPolyline_AZ_8(3MLIB)	106
mllib_GraphicsDrawPolyline_G_8(3MLIB)	107
mllib_GraphicsDrawPolyline_GZ_8(3MLIB)	108
mllib_GraphicsDrawPolyline_X_8(3MLIB)	109
mllib_GraphicsDrawPolyline_Z_8(3MLIB)	110
mllib_GraphicsDrawPolypoint_8(3MLIB)	111
mllib_GraphicsDrawPolypoint_X_8(3MLIB)	112
mllib_GraphicsDrawRectangle_8(3MLIB)	113
mllib_GraphicsDrawRectangle_X_8(3MLIB)	114
mllib_GraphicsDrawTriangle_8(3MLIB)	115
mllib_GraphicsDrawTriangle_A_8(3MLIB)	116
mllib_GraphicsDrawTriangle_AG_8(3MLIB)	117
mllib_GraphicsDrawTriangle_AGZ_8(3MLIB)	119
mllib_GraphicsDrawTriangle_AZ_8(3MLIB)	121
mllib_GraphicsDrawTriangleFanSet_8(3MLIB)	123
mllib_GraphicsDrawTriangleFanSet_A_8(3MLIB)	124
mllib_GraphicsDrawTriangleFanSet_AG_8(3MLIB)	125
mllib_GraphicsDrawTriangleFanSet_AGZ_8(3MLIB)	126
mllib_GraphicsDrawTriangleFanSet_AZ_8(3MLIB)	127
mllib_GraphicsDrawTriangleFanSet_G_8(3MLIB)	128
mllib_GraphicsDrawTriangleFanSet_GZ_8(3MLIB)	129
mllib_GraphicsDrawTriangleFanSet_X_8(3MLIB)	130
mllib_GraphicsDrawTriangleFanSet_Z_8(3MLIB)	131
mllib_GraphicsDrawTriangle_G_8(3MLIB)	132
mllib_GraphicsDrawTriangle_GZ_8(3MLIB)	134
mllib_GraphicsDrawTriangleSet_8(3MLIB)	136
mllib_GraphicsDrawTriangleSet_A_8(3MLIB)	137
mllib_GraphicsDrawTriangleSet_AG_8(3MLIB)	138
mllib_GraphicsDrawTriangleSet_AGZ_8(3MLIB)	139
mllib_GraphicsDrawTriangleSet_AZ_8(3MLIB)	140
mllib_GraphicsDrawTriangleSet_G_8(3MLIB)	141
mllib_GraphicsDrawTriangleSet_GZ_8(3MLIB)	142
mllib_GraphicsDrawTriangleSet_X_8(3MLIB)	143
mllib_GraphicsDrawTriangleSet_Z_8(3MLIB)	144
mllib_GraphicsDrawTriangleStripSet_8(3MLIB)	145

mllib_GraphicsDrawTriangleStripSet_A_8(3MLIB)	146
mllib_GraphicsDrawTriangleStripSet_AG_8(3MLIB)	147
mllib_GraphicsDrawTriangleStripSet_AGZ(3MLIB)	148
mllib_GraphicsDrawTriangleStripSet_AZ_8(3MLIB)	150
mllib_GraphicsDrawTriangleStripSet_G_8(3MLIB)	151
mllib_GraphicsDrawTriangleStripSet_GZ_8(3MLIB)	152
mllib_GraphicsDrawTriangleStripSet_X_8(3MLIB)	153
mllib_GraphicsDrawTriangleStripSet_Z_8(3MLIB)	154
mllib_GraphicsDrawTriangle_X_8(3MLIB)	155
mllib_GraphicsDrawTriangle_Z_8(3MLIB)	157
mllib_GraphicsFillArc_8(3MLIB)	159
mllib_GraphicsFillArc_A_8(3MLIB)	160
mllib_GraphicsFillArc_X_8(3MLIB)	161
mllib_GraphicsFillCircle_8(3MLIB)	162
mllib_GraphicsFillCircle_A_8(3MLIB)	163
mllib_GraphicsFillCircle_X_8(3MLIB)	164
mllib_GraphicsFillEllipse_8(3MLIB)	165
mllib_GraphicsFillEllipse_A_8(3MLIB)	166
mllib_GraphicsFillEllipse_X_8(3MLIB)	167
mllib_GraphicsFillPolygon_8(3MLIB)	168
mllib_GraphicsFillPolygon_A_8(3MLIB)	169
mllib_GraphicsFillPolygon_AG_8(3MLIB)	170
mllib_GraphicsFillPolygon_AGZ_8(3MLIB)	171
mllib_GraphicsFillPolygon_AZ_8(3MLIB)	172
mllib_GraphicsFillPolygon_G_8(3MLIB)	173
mllib_GraphicsFillPolygon_GZ_8(3MLIB)	174
mllib_GraphicsFillPolygon_X_8(3MLIB)	175
mllib_GraphicsFillPolygon_Z_8(3MLIB)	176
mllib_GraphicsFillRectangle_8(3MLIB)	177
mllib_GraphicsFillRectangle_X_8(3MLIB)	178
mllib_GraphicsFillTriangle_8(3MLIB)	179
mllib_GraphicsFillTriangle_A_8(3MLIB)	180
mllib_GraphicsFillTriangle_AG_8(3MLIB)	181
mllib_GraphicsFillTriangle_AGZ_8(3MLIB)	183
mllib_GraphicsFillTriangle_AZ_8(3MLIB)	185
mllib_GraphicsFillTriangleFanSet_8(3MLIB)	187
mllib_GraphicsFillTriangleFanSet_A_8(3MLIB)	188
mllib_GraphicsFillTriangleFanSet_AG_8(3MLIB)	189

mllib_GraphicsFillTriangleFanSet_AGZ_8(3MLIB)	190
mllib_GraphicsFillTriangleFanSet_AZ_8(3MLIB)	191
mllib_GraphicsFillTriangleFanSet_G_8(3MLIB)	192
mllib_GraphicsFillTriangleFanSet_GZ_8(3MLIB)	193
mllib_GraphicsFillTriangleFanSet_X_8(3MLIB)	194
mllib_GraphicsFillTriangleFanSet_Z_8(3MLIB)	195
mllib_GraphicsFillTriangle_G_8(3MLIB)	196
mllib_GraphicsFillTriangle_GZ_8(3MLIB)	198
mllib_GraphicsFillTriangleSet_8(3MLIB)	200
mllib_GraphicsFillTriangleSet_A_8(3MLIB)	201
mllib_GraphicsFillTriangleSet_AG_8(3MLIB)	202
mllib_GraphicsFillTriangleSet_AGZ_8(3MLIB)	203
mllib_GraphicsFillTriangleSet_AZ_8(3MLIB)	204
mllib_GraphicsFillTriangleSet_G_8(3MLIB)	205
mllib_GraphicsFillTriangleSet_GZ_8(3MLIB)	206
mllib_GraphicsFillTriangleSet_X_8(3MLIB)	207
mllib_GraphicsFillTriangleSet_Z_8(3MLIB)	208
mllib_GraphicsFillTriangleStripSet_8(3MLIB)	209
mllib_GraphicsFillTriangleStripSet_A_8(3MLIB)	210
mllib_GraphicsFillTriangleStripSet_AG_8(3MLIB)	211
mllib_GraphicsFillTriangleStripSet_AGZ(3MLIB)	212
mllib_GraphicsFillTriangleStripSet_AZ_8(3MLIB)	213
mllib_GraphicsFillTriangleStripSet_G_8(3MLIB)	214
mllib_GraphicsFillTriangleStripSet_GZ_8(3MLIB)	215
mllib_GraphicsFillTriangleStripSet_X_8(3MLIB)	216
mllib_GraphicsFillTriangleStripSet_Z_8(3MLIB)	217
mllib_GraphicsFillTriangle_X_8(3MLIB)	218
mllib_GraphicsFillTriangle_Z_8(3MLIB)	220
mllib_GraphicsFloodFill_8(3MLIB)	222
mllib_ImageAbs(3MLIB)	223
mllib_ImageAbs_Fp(3MLIB)	224
mllib_ImageAbs_Fp_Inp(3MLIB)	225
mllib_ImageAbs_Inp(3MLIB)	226
mllib_ImageAdd(3MLIB)	227
mllib_ImageAdd_Fp(3MLIB)	228
mllib_ImageAdd_Fp_Inp(3MLIB)	229
mllib_ImageAdd_Inp(3MLIB)	230
mllib_ImageAffine(3MLIB)	231

mlib_ImageAffine_Fp(3MLIB) 233
 mlib_ImageAffineIndex(3MLIB) 235
 mlib_ImageAffineTable(3MLIB) 237
 mlib_ImageAffineTable_Fp(3MLIB) 239
 mlib_ImageAffineTransform(3MLIB) 241
 mlib_ImageAffineTransform_Fp(3MLIB) 243
 mlib_ImageAffineTransformIndex(3MLIB) 245
 mlib_ImageAnd(3MLIB) 247
 mlib_ImageAnd_Inp(3MLIB) 248
 mlib_ImageAndNot1_Inp(3MLIB) 249
 mlib_ImageAndNot2_Inp(3MLIB) 250
 mlib_ImageAndNot(3MLIB) 251
 mlib_ImageAutoCorrel(3MLIB) 252
 mlib_ImageAutoCorrel_Fp(3MLIB) 253
 mlib_ImageAve(3MLIB) 254
 mlib_ImageAve_Fp(3MLIB) 255
 mlib_ImageAve_Fp_Inp(3MLIB) 256
 mlib_ImageAve_Inp(3MLIB) 257
 mlib_ImageBlend1_Fp_Inp(3MLIB) 258
 mlib_ImageBlend1_Inp(3MLIB) 259
 mlib_ImageBlend2_Fp_Inp(3MLIB) 260
 mlib_ImageBlend2_Inp(3MLIB) 261
 mlib_ImageBlend(3MLIB) 262
 mlib_ImageBlend_BSRC1_BSRC2(3MLIB) 263
 mlib_ImageBlend_BSRC1_BSRC2_Inp(3MLIB) 266
 mlib_ImageBlendColor(3MLIB) 269
 mlib_ImageBlendColor_Fp(3MLIB) 270
 mlib_ImageBlendColor_Fp_Inp(3MLIB) 271
 mlib_ImageBlendColor_Inp(3MLIB) 272
 mlib_ImageBlend_Fp(3MLIB) 273
 mlib_ImageBlendMulti(3MLIB) 274
 mlib_ImageBlendMulti_Fp(3MLIB) 276
 mlib_ImageBlendRGBA2ARGB(3MLIB) 278
 mlib_ImageBlendRGBA2BGRA(3MLIB) 279
 mlib_ImageChannelCopy(3MLIB) 280
 mlib_ImageChannelExtract(3MLIB) 281
 mlib_ImageChannelInsert(3MLIB) 282
 mlib_ImageChannelMerge(3MLIB) 283

mllib_ImageChannelSplit(3MLIB) 284
mllib_ImageClear(3MLIB) 285
mllib_ImageClearEdge(3MLIB) 286
mllib_ImageClearEdge_Fp(3MLIB) 287
mllib_ImageClear_Fp(3MLIB) 288
mllib_ImageColorConvert1(3MLIB) 289
mllib_ImageColorConvert1_Fp(3MLIB) 290
mllib_ImageColorConvert2(3MLIB) 291
mllib_ImageColorConvert2_Fp(3MLIB) 292
mllib_ImageColorDitherFree(3MLIB) 293
mllib_ImageColorDitherInit(3MLIB) 294
mllib_ImageColorErrorDiffusion3x3(3MLIB) 297
mllib_ImageColorErrorDiffusionMxN(3MLIB) 298
mllib_ImageColorHSL2RGB(3MLIB) 300
mllib_ImageColorHSL2RGB_Fp(3MLIB) 302
mllib_ImageColorHSV2RGB(3MLIB) 303
mllib_ImageColorHSV2RGB_Fp(3MLIB) 305
mllib_ImageColorOrderedDither8x8(3MLIB) 306
mllib_ImageColorOrderedDitherMxN(3MLIB) 307
mllib_ImageColorRGB2CIEMono(3MLIB) 309
mllib_ImageColorRGB2CIEMono_Fp(3MLIB) 310
mllib_ImageColorRGB2HSL(3MLIB) 311
mllib_ImageColorRGB2HSL_Fp(3MLIB) 313
mllib_ImageColorRGB2HSV(3MLIB) 314
mllib_ImageColorRGB2HSV_Fp(3MLIB) 316
mllib_ImageColorRGB2Mono(3MLIB) 317
mllib_ImageColorRGB2Mono_Fp(3MLIB) 318
mllib_ImageColorRGB2XYZ(3MLIB) 319
mllib_ImageColorRGB2XYZ_Fp(3MLIB) 320
mllib_ImageColorRGB2YCC(3MLIB) 321
mllib_ImageColorRGB2YCC_Fp(3MLIB) 322
mllib_ImageColorTrue2Index(3MLIB) 323
mllib_ImageColorTrue2IndexFree(3MLIB) 324
mllib_ImageColorTrue2IndexInit(3MLIB) 325
mllib_ImageColorXYZ2RGB(3MLIB) 327
mllib_ImageColorXYZ2RGB_Fp(3MLIB) 328
mllib_ImageColorYCC2RGB(3MLIB) 329
mllib_ImageColorYCC2RGB_Fp(3MLIB) 330

mllib_ImageComposite(3MLIB)	331
mllib_ImageComposite_Inp(3MLIB)	333
mllib_ImageConstAdd(3MLIB)	335
mllib_ImageConstAdd_Fp(3MLIB)	336
mllib_ImageConstAdd_Fp_Inp(3MLIB)	337
mllib_ImageConstAdd_Inp(3MLIB)	338
mllib_ImageConstAnd(3MLIB)	339
mllib_ImageConstAnd_Inp(3MLIB)	340
mllib_ImageConstAndNot(3MLIB)	341
mllib_ImageConstAndNot_Inp(3MLIB)	342
mllib_ImageConstDiv(3MLIB)	343
mllib_ImageConstDiv_Fp(3MLIB)	344
mllib_ImageConstDiv_Fp_Inp(3MLIB)	345
mllib_ImageConstDiv_Inp(3MLIB)	346
mllib_ImageConstDivShift(3MLIB)	347
mllib_ImageConstDivShift_Inp(3MLIB)	348
mllib_ImageConstMul(3MLIB)	349
mllib_ImageConstMul_Fp(3MLIB)	350
mllib_ImageConstMul_Fp_Inp(3MLIB)	351
mllib_ImageConstMul_Inp(3MLIB)	352
mllib_ImageConstMulShift(3MLIB)	353
mllib_ImageConstMulShift_Inp(3MLIB)	354
mllib_ImageConstNotAnd(3MLIB)	355
mllib_ImageConstNotAnd_Inp(3MLIB)	356
mllib_ImageConstNotOr(3MLIB)	357
mllib_ImageConstNotOr_Inp(3MLIB)	358
mllib_ImageConstNotXor(3MLIB)	359
mllib_ImageConstNotXor_Inp(3MLIB)	360
mllib_ImageConstOr(3MLIB)	361
mllib_ImageConstOr_Inp(3MLIB)	362
mllib_ImageConstOrNot(3MLIB)	363
mllib_ImageConstOrNot_Inp(3MLIB)	364
mllib_ImageConstSub(3MLIB)	365
mllib_ImageConstSub_Fp(3MLIB)	366
mllib_ImageConstSub_Fp_Inp(3MLIB)	367
mllib_ImageConstSub_Inp(3MLIB)	368
mllib_ImageConstXor(3MLIB)	369
mllib_ImageConstXor_Inp(3MLIB)	370

mllib_ImageConv2x2(3MLIB) 371
mllib_ImageConv2x2_Fp(3MLIB) 373
mllib_ImageConv2x2Index(3MLIB) 375
mllib_ImageConv3x3(3MLIB) 377
mllib_ImageConv3x3_Fp(3MLIB) 379
mllib_ImageConv3x3Index(3MLIB) 381
mllib_ImageConv4x4(3MLIB) 383
mllib_ImageConv4x4_Fp(3MLIB) 385
mllib_ImageConv4x4Index(3MLIB) 387
mllib_ImageConv5x5(3MLIB) 389
mllib_ImageConv5x5_Fp(3MLIB) 391
mllib_ImageConv5x5Index(3MLIB) 393
mllib_ImageConv7x7(3MLIB) 395
mllib_ImageConv7x7_Fp(3MLIB) 397
mllib_ImageConv7x7Index(3MLIB) 399
mllib_ImageConvKernelConvert(3MLIB) 401
mllib_ImageConvMxN(3MLIB) 403
mllib_ImageConvMxN_Fp(3MLIB) 405
mllib_ImageConvMxNIndex(3MLIB) 407
mllib_ImageConvolveMxN(3MLIB) 409
mllib_ImageConvolveMxN_Fp(3MLIB) 411
mllib_ImageCopy(3MLIB) 413
mllib_ImageCopyArea(3MLIB) 414
mllib_ImageCopyMask(3MLIB) 415
mllib_ImageCopyMask_Fp(3MLIB) 416
mllib_ImageCopySubimage(3MLIB) 417
mllib_ImageCreate(3MLIB) 418
mllib_ImageCreateStruct(3MLIB) 419
mllib_ImageCreateSubimage(3MLIB) 420
mllib_ImageCrossCorrel(3MLIB) 421
mllib_ImageCrossCorrel_Fp(3MLIB) 422
mllib_ImageDataTypeConvert(3MLIB) 423
mllib_ImageDelete(3MLIB) 426
mllib_ImageDilate4(3MLIB) 427
mllib_ImageDilate4_Fp(3MLIB) 428
mllib_ImageDilate8(3MLIB) 429
mllib_ImageDilate8_Fp(3MLIB) 430
mllib_ImageDiv1_Fp_Inp(3MLIB) 431

mllib_ImageDiv2_Fp_Inp(3MLIB) 432
mllib_ImageDivAlpha(3MLIB) 433
mllib_ImageDivAlpha_Fp(3MLIB) 435
mllib_ImageDivAlpha_Fp_Inp(3MLIB) 436
mllib_ImageDivAlpha_Inp(3MLIB) 437
mllib_ImageDivConstShift(3MLIB) 439
mllib_ImageDivConstShift_Inp(3MLIB) 440
mllib_ImageDiv_Fp(3MLIB) 441
mllib_ImageDivShift1_Inp(3MLIB) 442
mllib_ImageDivShift2_Inp(3MLIB) 443
mllib_ImageDivShift(3MLIB) 444
mllib_ImageErode4(3MLIB) 445
mllib_ImageErode4_Fp(3MLIB) 446
mllib_ImageErode8(3MLIB) 447
mllib_ImageErode8_Fp(3MLIB) 448
mllib_ImageExp(3MLIB) 449
mllib_ImageExp_Fp(3MLIB) 450
mllib_ImageExp_Fp_Inp(3MLIB) 451
mllib_ImageExp_Inp(3MLIB) 452
mllib_ImageExtrema2(3MLIB) 453
mllib_ImageExtremaLocations(3MLIB) 455
mllib_ImageFilteredSubsample(3MLIB) 458
mllib_ImageFlipAntiDiag(3MLIB) 461
mllib_ImageFlipAntiDiag_Fp(3MLIB) 462
mllib_ImageFlipMainDiag(3MLIB) 463
mllib_ImageFlipMainDiag_Fp(3MLIB) 464
mllib_ImageFlipX(3MLIB) 465
mllib_ImageFlipX_Fp(3MLIB) 466
mllib_ImageFlipY(3MLIB) 467
mllib_ImageFlipY_Fp(3MLIB) 468
mllib_ImageFourierTransform(3MLIB) 469
mllib_ImageGetBitOffset(3MLIB) 471
mllib_ImageGetChannels(3MLIB) 472
mllib_ImageGetData(3MLIB) 473
mllib_ImageGetFlags(3MLIB) 474
mllib_ImageGetFormat(3MLIB) 475
mllib_ImageGetHeight(3MLIB) 476
mllib_ImageGetPaddings(3MLIB) 477

mlib_ImageGetStride(3MLIB) 478
 mlib_ImageGetType(3MLIB) 479
 mlib_ImageGetWidth(3MLIB) 480
 mlib_ImageGradient3x3(3MLIB) 481
 mlib_ImageGradient3x3_Fp(3MLIB) 483
 mlib_ImageGradientMxN(3MLIB) 485
 mlib_ImageGradientMxN_Fp(3MLIB) 487
 mlib_ImageGridWarp(3MLIB) 489
 mlib_ImageGridWarp_Fp(3MLIB) 492
 mlib_ImageGridWarpTable(3MLIB) 495
 mlib_ImageGridWarpTable_Fp(3MLIB) 498
 mlib_ImageHistogram2(3MLIB) 501
 mlib_ImageHistogram(3MLIB) 503
 mlib_ImageInterpTableCreate(3MLIB) 504
 mlib_ImageInterpTableDelete(3MLIB) 506
 mlib_ImageInvert(3MLIB) 507
 mlib_ImageInvert_Fp(3MLIB) 508
 mlib_ImageInvert_Fp_Inp(3MLIB) 509
 mlib_ImageInvert_Inp(3MLIB) 510
 mlib_ImageIsNotAligned2(3MLIB) 511
 mlib_ImageIsNotAligned4(3MLIB) 512
 mlib_ImageIsNotAligned64(3MLIB) 513
 mlib_ImageIsNotAligned8(3MLIB) 514
 mlib_ImageIsNotHeight2X(3MLIB) 515
 mlib_ImageIsNotHeight4X(3MLIB) 516
 mlib_ImageIsNotHeight8X(3MLIB) 517
 mlib_ImageIsNotOneDvector(3MLIB) 518
 mlib_ImageIsNotStride8X(3MLIB) 519
 mlib_ImageIsNotWidth2X(3MLIB) 520
 mlib_ImageIsNotWidth4X(3MLIB) 521
 mlib_ImageIsNotWidth8X(3MLIB) 522
 mlib_ImageIsUserAllocated(3MLIB) 523
 mlib_ImageLog(3MLIB) 524
 mlib_ImageLog_Fp(3MLIB) 525
 mlib_ImageLog_Fp_Inp(3MLIB) 526
 mlib_ImageLog_Inp(3MLIB) 527
 mlib_ImageLookUp2(3MLIB) 528
 mlib_ImageLookUp(3MLIB) 530

mlib_ImageLookUp_Inp(3MLIB) 532
 mlib_ImageLookUpMask(3MLIB) 533
 mlib_ImageMax(3MLIB) 535
 mlib_ImageMaxFilter3x3(3MLIB) 536
 mlib_ImageMaxFilter3x3_Fp(3MLIB) 538
 mlib_ImageMaxFilter5x5(3MLIB) 540
 mlib_ImageMaxFilter5x5_Fp(3MLIB) 542
 mlib_ImageMaxFilter7x7(3MLIB) 544
 mlib_ImageMaxFilter7x7_Fp(3MLIB) 546
 mlib_ImageMax_Fp(3MLIB) 548
 mlib_ImageMax_Fp_Inp(3MLIB) 549
 mlib_ImageMaximum(3MLIB) 550
 mlib_ImageMaximum_Fp(3MLIB) 551
 mlib_ImageMax_Inp(3MLIB) 552
 mlib_ImageMean(3MLIB) 553
 mlib_ImageMean_Fp(3MLIB) 554
 mlib_ImageMedianFilter3x3(3MLIB) 555
 mlib_ImageMedianFilter3x3_Fp(3MLIB) 557
 mlib_ImageMedianFilter3x3_US(3MLIB) 559
 mlib_ImageMedianFilter5x5(3MLIB) 561
 mlib_ImageMedianFilter5x5_Fp(3MLIB) 563
 mlib_ImageMedianFilter5x5_US(3MLIB) 565
 mlib_ImageMedianFilter7x7(3MLIB) 567
 mlib_ImageMedianFilter7x7_Fp(3MLIB) 569
 mlib_ImageMedianFilter7x7_US(3MLIB) 571
 mlib_ImageMedianFilterMxN(3MLIB) 573
 mlib_ImageMedianFilterMxN_Fp(3MLIB) 575
 mlib_ImageMedianFilterMxN_US(3MLIB) 577
 mlib_ImageMin(3MLIB) 579
 mlib_ImageMinFilter3x3(3MLIB) 580
 mlib_ImageMinFilter3x3_Fp(3MLIB) 582
 mlib_ImageMinFilter5x5(3MLIB) 584
 mlib_ImageMinFilter5x5_Fp(3MLIB) 586
 mlib_ImageMinFilter7x7(3MLIB) 588
 mlib_ImageMinFilter7x7_Fp(3MLIB) 590
 mlib_ImageMin_Fp(3MLIB) 592
 mlib_ImageMin_Fp_Inp(3MLIB) 593
 mlib_ImageMinimum(3MLIB) 594

mllib_ImageMinimum_Fp(3MLIB) 595
mllib_ImageMin_Inp(3MLIB) 596
mllib_ImageMoment2(3MLIB) 597
mllib_ImageMoment2_Fp(3MLIB) 598
mllib_ImageMulAlpha(3MLIB) 599
mllib_ImageMulAlpha_Fp(3MLIB) 600
mllib_ImageMulAlpha_Fp_Inp(3MLIB) 601
mllib_ImageMulAlpha_Inp(3MLIB) 602
mllib_ImageMul_Fp(3MLIB) 603
mllib_ImageMul_Fp_Inp(3MLIB) 604
mllib_ImageMulShift(3MLIB) 605
mllib_ImageMulShift_Inp(3MLIB) 606
mllib_ImageNot(3MLIB) 607
mllib_ImageNotAnd(3MLIB) 608
mllib_ImageNotAnd_Inp(3MLIB) 609
mllib_ImageNot_Inp(3MLIB) 610
mllib_ImageNotOr(3MLIB) 611
mllib_ImageNotOr_Inp(3MLIB) 612
mllib_ImageNotXor(3MLIB) 613
mllib_ImageNotXor_Inp(3MLIB) 614
mllib_ImageOr(3MLIB) 615
mllib_ImageOr_Inp(3MLIB) 616
mllib_ImageOrNot1_Inp(3MLIB) 617
mllib_ImageOrNot2_Inp(3MLIB) 618
mllib_ImageOrNot(3MLIB) 619
mllib_ImagePolynomialWarp(3MLIB) 620
mllib_ImagePolynomialWarp_Fp(3MLIB) 623
mllib_ImagePolynomialWarpTable(3MLIB) 626
mllib_ImagePolynomialWarpTable_Fp(3MLIB) 629
mllib_ImageRankFilter3x3(3MLIB) 632
mllib_ImageRankFilter3x3_Fp(3MLIB) 634
mllib_ImageRankFilter3x3_US(3MLIB) 636
mllib_ImageRankFilter5x5(3MLIB) 638
mllib_ImageRankFilter5x5_Fp(3MLIB) 640
mllib_ImageRankFilter5x5_US(3MLIB) 642
mllib_ImageRankFilter7x7(3MLIB) 644
mllib_ImageRankFilter7x7_Fp(3MLIB) 646
mllib_ImageRankFilter7x7_US(3MLIB) 648

mlib_ImageRankFilterMxN(3MLIB) 650
 mlib_ImageRankFilterMxN_Fp(3MLIB) 652
 mlib_ImageRankFilterMxN_US(3MLIB) 654
 mlib_ImageReformat(3MLIB) 656
 mlib_ImageReplaceColor(3MLIB) 659
 mlib_ImageReplaceColor_Fp(3MLIB) 660
 mlib_ImageReplaceColor_Fp_Inp(3MLIB) 661
 mlib_ImageReplaceColor_Inp(3MLIB) 662
 mlib_ImageRotate180(3MLIB) 663
 mlib_ImageRotate180_Fp(3MLIB) 664
 mlib_ImageRotate270(3MLIB) 665
 mlib_ImageRotate270_Fp(3MLIB) 666
 mlib_ImageRotate(3MLIB) 667
 mlib_ImageRotate90(3MLIB) 669
 mlib_ImageRotate90_Fp(3MLIB) 670
 mlib_ImageRotate_Fp(3MLIB) 671
 mlib_ImageRotateIndex(3MLIB) 673
 mlib_ImageScalarBlend(3MLIB) 675
 mlib_ImageScalarBlend_Fp(3MLIB) 676
 mlib_ImageScalarBlend_Fp_Inp(3MLIB) 677
 mlib_ImageScalarBlend_Inp(3MLIB) 678
 mlib_ImageScale2(3MLIB) 679
 mlib_ImageScale2_Inp(3MLIB) 681
 mlib_ImageScale(3MLIB) 682
 mlib_ImageScale_Fp(3MLIB) 684
 mlib_ImageScale_Fp_Inp(3MLIB) 686
 mlib_ImageScale_Inp(3MLIB) 687
 mlib_ImageSConv3x3(3MLIB) 688
 mlib_ImageSConv3x3_Fp(3MLIB) 690
 mlib_ImageSConv5x5(3MLIB) 692
 mlib_ImageSConv5x5_Fp(3MLIB) 694
 mlib_ImageSConv7x7(3MLIB) 696
 mlib_ImageSConv7x7_Fp(3MLIB) 698
 mlib_ImageSConvKernelConvert(3MLIB) 700
 mlib_ImageSetFormat(3MLIB) 701
 mlib_ImageSetPaddings(3MLIB) 702
 mlib_ImageSobel(3MLIB) 704
 mlib_ImageSqr_Fp(3MLIB) 706

mllib_ImageSqr_Fp_Inp(3MLIB) 707
mllib_ImageSqrShift(3MLIB) 708
mllib_ImageSqrShift_Inp(3MLIB) 709
mllib_ImageStdDev(3MLIB) 710
mllib_ImageStdDev_Fp(3MLIB) 711
mllib_ImageSub1_Fp_Inp(3MLIB) 712
mllib_ImageSub1_Inp(3MLIB) 713
mllib_ImageSub2_Fp_Inp(3MLIB) 714
mllib_ImageSub2_Inp(3MLIB) 715
mllib_ImageSub(3MLIB) 716
mllib_ImageSub_Fp(3MLIB) 717
mllib_ImageSubsampleAverage(3MLIB) 718
mllib_ImageSubsampleBinaryToGray(3MLIB) 720
mllib_ImageTestFlags(3MLIB) 722
mllib_ImageThresh1(3MLIB) 723
mllib_ImageThresh1_Fp(3MLIB) 725
mllib_ImageThresh1_Fp_Inp(3MLIB) 727
mllib_ImageThresh1_Inp(3MLIB) 728
mllib_ImageThresh2(3MLIB) 729
mllib_ImageThresh2_Fp(3MLIB) 730
mllib_ImageThresh2_Fp_Inp(3MLIB) 731
mllib_ImageThresh2_Inp(3MLIB) 732
mllib_ImageThresh3(3MLIB) 733
mllib_ImageThresh3_Fp(3MLIB) 734
mllib_ImageThresh3_Fp_Inp(3MLIB) 735
mllib_ImageThresh3_Inp(3MLIB) 736
mllib_ImageThresh4(3MLIB) 737
mllib_ImageThresh4_Fp(3MLIB) 739
mllib_ImageThresh4_Fp_Inp(3MLIB) 741
mllib_ImageThresh4_Inp(3MLIB) 743
mllib_ImageThresh5(3MLIB) 745
mllib_ImageThresh5_Fp(3MLIB) 747
mllib_ImageThresh5_Fp_Inp(3MLIB) 749
mllib_ImageThresh5_Inp(3MLIB) 750
mllib_ImageXor(3MLIB) 751
mllib_ImageXor_Inp(3MLIB) 752
mllib_ImageXProj(3MLIB) 753
mllib_ImageXProj_Fp(3MLIB) 754

mlib_ImageYProj(3MLIB) 755
 mlib_ImageYProj_Fp(3MLIB) 756
 mlib_ImageZoom(3MLIB) 757
 mlib_ImageZoomBlend(3MLIB) 759
 mlib_ImageZoom_Fp(3MLIB) 761
 mlib_ImageZoomIn2X(3MLIB) 763
 mlib_ImageZoomIn2X_Fp(3MLIB) 764
 mlib_ImageZoomIn2XIndex(3MLIB) 766
 mlib_ImageZoomIndex(3MLIB) 768
 mlib_ImageZoomOut2X(3MLIB) 770
 mlib_ImageZoomOut2X_Fp(3MLIB) 771
 mlib_ImageZoomOut2XIndex(3MLIB) 773
 mlib_ImageZoomTranslate(3MLIB) 775
 mlib_ImageZoomTranslateBlend(3MLIB) 777
 mlib_ImageZoomTranslate_Fp(3MLIB) 780
 mlib_ImageZoomTranslateTable(3MLIB) 782
 mlib_ImageZoomTranslateTableBlend(3MLIB) 784
 mlib_ImageZoomTranslateTable_Fp(3MLIB) 787
 mlib_ImageZoomTranslateToGray(3MLIB) 789

Multimedia Library Functions Part Two 791

mlib_malloc(3MLIB) 792
 mlib_MatrixAddS_U8_Mod(3MLIB) 793
 mlib_MatrixAddS_U8_U8_Mod(3MLIB) 795
 mlib_MatrixAdd_U8_Mod(3MLIB) 798
 mlib_MatrixAdd_U8_U8_Mod(3MLIB) 800
 mlib_MatrixMaximumMag_U8C(3MLIB) 803
 mlib_MatrixMaximum_U8(3MLIB) 804
 mlib_MatrixMinimumMag_U8C(3MLIB) 805
 mlib_MatrixMinimum_U8(3MLIB) 806
 mlib_MatrixMulShift_S16_S16_Mod(3MLIB) 807
 mlib_MatrixMulSShift_U8_Mod(3MLIB) 809
 mlib_MatrixMulSShift_U8_U8_Mod(3MLIB) 811
 mlib_MatrixMulS_U8_Mod(3MLIB) 814
 mlib_MatrixMulS_U8_U8_Mod(3MLIB) 816
 mlib_MatrixMul_U8_U8_Mod(3MLIB) 819
 mlib_MatrixScale_U8_Mod(3MLIB) 823
 mlib_MatrixScale_U8_U8_Mod(3MLIB) 825

mlib_MatrixSubS_U8_Mod(3MLIB) 829
 mlib_MatrixSubS_U8_U8_Mod(3MLIB) 831
 mlib_MatrixSub_U8_Mod(3MLIB) 834
 mlib_MatrixSub_U8_U8_Mod(3MLIB) 836
 mlib_MatrixTranspose_U8(3MLIB) 839
 mlib_MatrixTranspose_U8_U8(3MLIB) 841
 mlib_MatrixUnit_U8(3MLIB) 843
 mlib_memcpy(3MLIB) 845
 mlib_memmove(3MLIB) 846
 mlib_memset(3MLIB) 847
 mlib_realloc(3MLIB) 848
 mlib_SignalADPCM2Bits2Linear(3MLIB) 849
 mlib_SignalADPCM3Bits2Linear(3MLIB) 850
 mlib_SignalADPCM4Bits2Linear(3MLIB) 851
 mlib_SignalADPCM5Bits2Linear(3MLIB) 852
 mlib_SignalADPCMFree(3MLIB) 853
 mlib_SignalADPCMInit(3MLIB) 854
 mlib_SignalALaw2Linear(3MLIB) 855
 mlib_SignalALaw2uLaw(3MLIB) 856
 mlib_SignalAutoCorrel_S16(3MLIB) 857
 mlib_SignalCepstral_F32(3MLIB) 859
 mlib_SignalCepstralFree_S16(3MLIB) 861
 mlib_SignalCepstralInit_S16(3MLIB) 862
 mlib_SignalCepstral_S16(3MLIB) 863
 mlib_SignalCepstral_S16_Adp(3MLIB) 865
 mlib_SignalConvertShift_F32_U8(3MLIB) 867
 mlib_SignalConvertShift_U8_S8_Sat(3MLIB) 869
 mlib_SignalConv_S16_S16_Sat(3MLIB) 873
 mlib_SignalCrossCorrel_S16(3MLIB) 875
 mlib_SignalDownSample_S16_S16(3MLIB) 877
 mlib_SignalDTWKScalar_F32(3MLIB) 879
 mlib_SignalDTWKScalarFree_S16(3MLIB) 883
 mlib_SignalDTWKScalarInit_F32(3MLIB) 884
 mlib_SignalDTWKScalarInit_S16(3MLIB) 885
 mlib_SignalDTWKScalarPath_S16(3MLIB) 886
 mlib_SignalDTWKScalar_S16(3MLIB) 890
 mlib_SignalDTWKVector_F32(3MLIB) 894
 mlib_SignalDTWKVectorFree_S16(3MLIB) 898

mllib_SignalDTWKVectorInit_F32(3MLIB)	899
mllib_SignalDTWKVectorInit_S16(3MLIB)	901
mllib_SignalDTWKVectorPath_S16(3MLIB)	903
mllib_SignalDTWKVector_S16(3MLIB)	907
mllib_SignalDTWScalar_F32(3MLIB)	911
mllib_SignalDTWScalarFree_S16(3MLIB)	915
mllib_SignalDTWScalarInit_F32(3MLIB)	916
mllib_SignalDTWScalarInit_S16(3MLIB)	917
mllib_SignalDTWScalarPath_F32(3MLIB)	918
mllib_SignalDTWScalarPath_S16(3MLIB)	922
mllib_SignalDTWScalar_S16(3MLIB)	926
mllib_SignalDTWVector_F32(3MLIB)	930
mllib_SignalDTWVectorFree_S16(3MLIB)	934
mllib_SignalDTWVectorInit_F32(3MLIB)	935
mllib_SignalDTWVectorInit_S16(3MLIB)	936
mllib_SignalDTWVectorPath_F32(3MLIB)	938
mllib_SignalDTWVectorPath_S16(3MLIB)	942
mllib_SignalDTWVector_S16(3MLIB)	946
mllib_SignalEmphasizeFree_S16_S16(3MLIB)	950
mllib_SignalEmphasizeInit_S16_S16(3MLIB)	951
mllib_SignalEmphasize_S16_S16_Sat(3MLIB)	952
mllib_SignalFFT_1(3MLIB)	954
mllib_SignalFFT_2(3MLIB)	957
mllib_SignalFFT_3(3MLIB)	960
mllib_SignalFFT_4(3MLIB)	963
mllib_SignalFFTW_1(3MLIB)	965
mllib_SignalFFTW_2(3MLIB)	968
mllib_SignalFFTW_3(3MLIB)	971
mllib_SignalFFTW_4(3MLIB)	974
mllib_SignalFIR_F32_F32(3MLIB)	977
mllib_SignalFIR_F32S_F32S(3MLIB)	978
mllib_SignalFIRFree_F32_F32(3MLIB)	979
mllib_SignalFIRFree_F32S_F32S(3MLIB)	980
mllib_SignalFIRFree_S16_S16(3MLIB)	981
mllib_SignalFIRInit_F32_F32(3MLIB)	982
mllib_SignalFIRInit_F32S_F32S(3MLIB)	983
mllib_SignalFIRInit_S16_S16(3MLIB)	984
mllib_SignalFIR_S16_S16_Sat(3MLIB)	985

mllib_SignalGaussNoise_F32(3MLIB)	986
mllib_SignalGaussNoiseFree_F32(3MLIB)	987
mllib_SignalGaussNoiseFree_S16(3MLIB)	988
mllib_SignalGaussNoiseInit_F32(3MLIB)	989
mllib_SignalGaussNoiseInit_S16(3MLIB)	990
mllib_SignalGaussNoise_S16(3MLIB)	991
mllib_SignalGenBartlett_F32(3MLIB)	992
mllib_SignalGenBartlett_S16(3MLIB)	993
mllib_SignalGenBlackman_F32(3MLIB)	994
mllib_SignalGenBlackman_S16(3MLIB)	995
mllib_SignalGenHamming_F32(3MLIB)	996
mllib_SignalGenHamming_S16(3MLIB)	997
mllib_SignalGenHanning_F32(3MLIB)	998
mllib_SignalGenHanning_S16(3MLIB)	999
mllib_SignalGenKaiser_F32(3MLIB)	1000
mllib_SignalGenKaiser_S16(3MLIB)	1001
mllib_SignalIFFT_1(3MLIB)	1002
mllib_SignalIFFT_2(3MLIB)	1005
mllib_SignalIFFT_3(3MLIB)	1008
mllib_SignalIFFT_4(3MLIB)	1011
mllib_SignalIFFTW_1(3MLIB)	1013
mllib_SignalIFFTW_2(3MLIB)	1016
mllib_SignalIFFTW_3(3MLIB)	1019
mllib_SignalIFFTW_4(3MLIB)	1022
mllib_SignalIIR_Biquad_S16_S16_Sat(3MLIB)	1025
mllib_SignalIIRFree_Biquad_F32_F32(3MLIB)	1027
mllib_SignalIIRFree_Biquad_S16_S16(3MLIB)	1028
mllib_SignalIIRFree_P4_F32_F32(3MLIB)	1029
mllib_SignalIIRFree_P4_S16_S16(3MLIB)	1030
mllib_SignalIIRInit_Biquad_F32_F32(3MLIB)	1031
mllib_SignalIIRInit_Biquad_S16_S16(3MLIB)	1032
mllib_SignalIIRInit_P4_F32_F32(3MLIB)	1033
mllib_SignalIIRInit_P4_S16_S16(3MLIB)	1034
mllib_SignalIIR_P4_S16_S16_Sat(3MLIB)	1035
mllib_SignalIMDCT_D64(3MLIB)	1038
mllib_SignalIMDCT_F32(3MLIB)	1039
mllib_SignalIMDCTSplit_D64(3MLIB)	1040
mllib_SignalIMDCTSplit_F32(3MLIB)	1041

mllib_SignalLimit(3MLIB)	1042
mllib_SignalLinear2ADPCM2Bits(3MLIB)	1044
mllib_SignalLinear2ADPCM3Bits(3MLIB)	1045
mllib_SignalLinear2ADPCM4Bits(3MLIB)	1046
mllib_SignalLinear2ADPCM5Bits(3MLIB)	1047
mllib_SignalLinear2ALaw(3MLIB)	1048
mllib_SignalLinear2uLaw(3MLIB)	1049
mllib_SignalLMSFilter_F32_F32(3MLIB)	1050
mllib_SignalLMSFilterFree_F32_F32(3MLIB)	1051
mllib_SignalLMSFilterFree_S16_S16(3MLIB)	1052
mllib_SignalLMSFilterInit_F32_F32(3MLIB)	1053
mllib_SignalLMSFilterInit_S16_S16(3MLIB)	1054
mllib_SignalLMSFilter_S16_S16_Sat(3MLIB)	1055
mllib_SignalLPC2Cepstral_F32(3MLIB)	1056
mllib_SignalLPC2Cepstral_S16(3MLIB)	1058
mllib_SignalLPC2Cepstral_S16_Adp(3MLIB)	1060
mllib_SignalLPC2LSP_F32(3MLIB)	1062
mllib_SignalLPC2LSP_S16(3MLIB)	1064
mllib_SignalLPCAutoCorrel_F32(3MLIB)	1066
mllib_SignalLPCAutoCorrelFree_S16(3MLIB)	1068
mllib_SignalLPCAutoCorrelGetEnergy_F32(3MLIB)	1069
mllib_SignalLPCAutoCorrelGetEnergy_S16(3MLIB)	1071
mllib_SignalLPCAutoCorrelGetPARCOR_F32(3MLIB)	1073
mllib_SignalLPCAutoCorrelGetPARCOR_S16(3MLIB)	1075
mllib_SignalLPCAutoCorrelInit_S16(3MLIB)	1077
mllib_SignalLPCAutoCorrel_S16(3MLIB)	1078
mllib_SignalLPCCovariance_F32(3MLIB)	1080
mllib_SignalLPCCovarianceFree_S16(3MLIB)	1082
mllib_SignalLPCCovarianceInit_S16(3MLIB)	1083
mllib_SignalLPCCovariance_S16(3MLIB)	1084
mllib_SignalLPCPerceptWeight_F32(3MLIB)	1086
mllib_SignalLPCPerceptWeightFree_S16(3MLIB)	1087
mllib_SignalLPCPerceptWeightInit_S16(3MLIB)	1088
mllib_SignalLPCPerceptWeight_S16(3MLIB)	1089
mllib_SignalLPCPitchAnalyze_F32(3MLIB)	1091
mllib_SignalLPCPitchAnalyze_S16(3MLIB)	1093
mllib_SignalLSP2LPC_F32(3MLIB)	1095
mllib_SignalLSP2LPC_S16(3MLIB)	1097

mllib_SignalMelCepstral_F32(3MLIB) 1099
mllib_SignalMelCepstralFree_S16(3MLIB) 1101
mllib_SignalMelCepstralInit_S16(3MLIB) 1102
mllib_SignalMelCepstral_S16(3MLIB) 1104
mllib_SignalMelCepstral_S16_Adp(3MLIB) 1106
mllib_SignalMerge_F32S_F32(3MLIB) 1108
mllib_SignalMerge_S16S_S16(3MLIB) 1109
mllib_SignalMulBartlett_F32(3MLIB) 1110
mllib_SignalMulBartlett_F32_F32(3MLIB) 1111
mllib_SignalMulBartlett_S16(3MLIB) 1112
mllib_SignalMulBartlett_S16_S16(3MLIB) 1113
mllib_SignalMulBlackman_F32(3MLIB) 1114
mllib_SignalMulBlackman_F32_F32(3MLIB) 1115
mllib_SignalMulBlackman_S16(3MLIB) 1116
mllib_SignalMulBlackman_S16_S16(3MLIB) 1117
mllib_SignalMul_F32(3MLIB) 1118
mllib_SignalMul_F32_F32(3MLIB) 1119
mllib_SignalMulHamming_F32(3MLIB) 1120
mllib_SignalMulHamming_F32_F32(3MLIB) 1121
mllib_SignalMulHamming_S16(3MLIB) 1122
mllib_SignalMulHamming_S16_S16(3MLIB) 1123
mllib_SignalMulHanning_F32(3MLIB) 1124
mllib_SignalMulHanning_F32_F32(3MLIB) 1125
mllib_SignalMulHanning_S16(3MLIB) 1126
mllib_SignalMulHanning_S16_S16(3MLIB) 1127
mllib_SignalMulKaiser_F32(3MLIB) 1128
mllib_SignalMulKaiser_F32_F32(3MLIB) 1129
mllib_SignalMulKaiser_S16(3MLIB) 1130
mllib_SignalMulKaiser_S16_S16(3MLIB) 1131
mllib_SignalMulRectangular_F32(3MLIB) 1132
mllib_SignalMulRectangular_F32_F32(3MLIB) 1133
mllib_SignalMulRectangular_S16(3MLIB) 1134
mllib_SignalMulRectangular_S16_S16(3MLIB) 1135
mllib_SignalMul_S16_S16_Sat(3MLIB) 1136
mllib_SignalMul_S16_Sat(3MLIB) 1137
mllib_SignalMulSAdd_F32(3MLIB) 1138
mllib_SignalMulSAdd_F32_F32(3MLIB) 1139
mllib_SignalMulSAdd_S16_S16_Sat(3MLIB) 1140

mlib_SignalMulSAdd_S16_Sat(3MLIB) 1141
 mlib_SignalMulS_F32(3MLIB) 1142
 mlib_SignalMulS_F32_F32(3MLIB) 1143
 mlib_SignalMulShift_S16_S16_Sat(3MLIB) 1144
 mlib_SignalMulShift_S16_Sat(3MLIB) 1145
 mlib_SignalMulS_S16_S16_Sat(3MLIB) 1146
 mlib_SignalMulS_S16_Sat(3MLIB) 1147
 mlib_SignalMulSShiftAdd_S16_S16_Sat(3MLIB) 1148
 mlib_SignalMulSShiftAdd_S16_Sat(3MLIB) 1149
 mlib_SignalMulSShift_S16_S16_Sat(3MLIB) 1150
 mlib_SignalMulSShift_S16_Sat(3MLIB) 1151
 mlib_SignalMulWindow_F32(3MLIB) 1152
 mlib_SignalMulWindow_F32_F32(3MLIB) 1153
 mlib_SignalMulWindow_F32S(3MLIB) 1154
 mlib_SignalMulWindow_F32S_F32S(3MLIB) 1155
 mlib_SignalMulWindow_S16(3MLIB) 1156
 mlib_SignalMulWindow_S16_S16(3MLIB) 1157
 mlib_SignalQuant2_S16_F32(3MLIB) 1158
 mlib_SignalQuant2_S16S_F32S(3MLIB) 1159
 mlib_SignalQuant_S16_F32(3MLIB) 1160
 mlib_SignalQuant_S16S_F32S(3MLIB) 1161
 mlib_SignalQuant_U8_F32(3MLIB) 1162
 mlib_SignalQuant_U8_S16(3MLIB) 1163
 mlib_SignalQuant_U8S_F32S(3MLIB) 1164
 mlib_SignalReSampleFIR_F32_F32(3MLIB) 1165
 mlib_SignalReSampleFIR_F32S_F32S(3MLIB) 1166
 mlib_SignalReSampleFIRFree_F32_F32(3MLIB) 1167
 mlib_SignalReSampleFIRFree_F32S_F32S(3MLIB) 1168
 mlib_SignalReSampleFIRFree_S16_S16(3MLIB) 1169
 mlib_SignalReSampleFIRInit_S16_S16(3MLIB) 1170
 mlib_SignalReSampleFIR_S16_S16_Sat(3MLIB) 1172
 mlib_SignalSineWave_F32(3MLIB) 1173
 mlib_SignalSineWaveFree_F32(3MLIB) 1174
 mlib_SignalSineWaveFree_S16(3MLIB) 1175
 mlib_SignalSineWaveInit_F32(3MLIB) 1176
 mlib_SignalSineWaveInit_S16(3MLIB) 1177
 mlib_SignalSineWave_S16(3MLIB) 1178
 mlib_SignalSplit_F32_F32S(3MLIB) 1179

mlib_SignalSplit_S16_S16S(3MLIB) 1180
 mlib_SignaluLaw2ALaw(3MLIB) 1181
 mlib_SignaluLaw2Linear(3MLIB) 1182
 mlib_SignalUpSampleFIR_F32_F32(3MLIB) 1183
 mlib_SignalUpSampleFIR_F32S_F32S(3MLIB) 1184
 mlib_SignalUpSampleFIRFree_F32_F32(3MLIB) 1185
 mlib_SignalUpSampleFIRFree_F32S_F32S(3MLIB) 1186
 mlib_SignalUpSampleFIRFree_S16_S16(3MLIB) 1187
 mlib_SignalUpSampleFIRInit_F32_F32(3MLIB) 1188
 mlib_SignalUpSampleFIRInit_F32S_F32S(3MLIB) 1189
 mlib_SignalUpSampleFIRInit_S16_S16(3MLIB) 1190
 mlib_SignalUpSampleFIR_S16_S16_Sat(3MLIB) 1191
 mlib_SignalUpSample_S16_S16(3MLIB) 1192
 mlib_SignalWhiteNoise_F32(3MLIB) 1194
 mlib_SignalWhiteNoiseFree_F32(3MLIB) 1195
 mlib_SignalWhiteNoiseFree_S16(3MLIB) 1196
 mlib_SignalWhiteNoiseInit_F32(3MLIB) 1197
 mlib_SignalWhiteNoiseInit_S16(3MLIB) 1198
 mlib_SignalWhiteNoise_S16(3MLIB) 1199
 mlib_VectorAddS_U8_Mod(3MLIB) 1200
 mlib_VectorAddS_U8_U8_Mod(3MLIB) 1202
 mlib_VectorAdd_U8_Mod(3MLIB) 1205
 mlib_VectorAdd_U8_U8_Mod(3MLIB) 1207
 mlib_VectorAng_U8C(3MLIB) 1210
 mlib_VectorConjRev_S8C_S8C_Sat(3MLIB) 1211
 mlib_VectorConj_S8C_S8C_Sat(3MLIB) 1212
 mlib_VectorConj_S8C_Sat(3MLIB) 1213
 mlib_VectorConjSymExt_S8C_S8C_Sat(3MLIB) 1214
 mlib_VectorConvert_U8_S8_Mod(3MLIB) 1216
 mlib_VectorCopy_U8(3MLIB) 1221
 mlib_VectorDistance_U8_Sat(3MLIB) 1223
 mlib_VectorDotProd_U8_Sat(3MLIB) 1224
 mlib_VectorMag_U8C(3MLIB) 1226
 mlib_VectorMaximumMag_U8C(3MLIB) 1227
 mlib_VectorMaximum_U8(3MLIB) 1228
 mlib_VectorMerge_U8C_U8(3MLIB) 1229
 mlib_VectorMinimumMag_U8C(3MLIB) 1230
 mlib_VectorMinimum_U8(3MLIB) 1231

mlib_VectorMulMShift_S16_S16_Mod(3MLIB) 1232
 mlib_VectorMulM_U8_U8_Mod(3MLIB) 1234
 mlib_VectorMulSAdd_U8_Mod(3MLIB) 1237
 mlib_VectorMulSAdd_U8_U8_Mod(3MLIB) 1239
 mlib_VectorMulShift_U8_Mod(3MLIB) 1242
 mlib_VectorMulShift_U8_U8_Mod(3MLIB) 1244
 mlib_VectorMulSShift_U8_Mod(3MLIB) 1246
 mlib_VectorMulSShift_U8_U8_Mod(3MLIB) 1248
 mlib_VectorMulS_U8_Mod(3MLIB) 1250
 mlib_VectorMulS_U8_U8_Mod(3MLIB) 1252
 mlib_VectorMul_U8_Mod(3MLIB) 1255
 mlib_VectorMul_U8_U8_Mod(3MLIB) 1257
 mlib_VectorNorm_U8_Sat(3MLIB) 1260
 mlib_VectorReverseByteOrder(3MLIB) 1261
 mlib_VectorReverseByteOrder_Inp(3MLIB) 1262
 mlib_VectorReverseByteOrder_S16(3MLIB) 1263
 mlib_VectorReverseByteOrder_S16_S16(3MLIB) 1265
 mlib_VectorScale_U8_Mod(3MLIB) 1267
 mlib_VectorScale_U8_U8_Mod(3MLIB) 1269
 mlib_VectorSet_U8(3MLIB) 1272
 mlib_VectorSplit_U8_U8C(3MLIB) 1274
 mlib_VectorSubS_U8_Mod(3MLIB) 1275
 mlib_VectorSubS_U8_U8_Mod(3MLIB) 1277
 mlib_VectorSub_U8_Mod(3MLIB) 1280
 mlib_VectorSub_U8_U8_Mod(3MLIB) 1282
 mlib_VectorSumAbsDiff_U8_Sat(3MLIB) 1285
 mlib_VectorSumAbs_U8_Sat(3MLIB) 1286
 mlib_VectorZero_U8(3MLIB) 1287
 mlib_version(3MLIB) 1288
 mlib_VideoAddBlock_U8_S16(3MLIB) 1289
 mlib_VideoColorABGR2JFIFYCC420(3MLIB) 1291
 mlib_VideoColorABGR2JFIFYCC422(3MLIB) 1292
 mlib_VideoColorABGR2JFIFYCC444(3MLIB) 1293
 mlib_VideoColorABGR2RGB(3MLIB) 1294
 mlib_VideoColorABGRint_to_ARGBint(3MLIB) 1295
 mlib_VideoColorARGB2JFIFYCC420(3MLIB) 1296
 mlib_VideoColorARGB2JFIFYCC422(3MLIB) 1297
 mlib_VideoColorARGB2JFIFYCC444(3MLIB) 1298

mlib_VideoColorARGB2RGB(3MLIB) 1299
 mlib_VideoColorBGRAint_to_ABGRint(3MLIB) 1300
 mlib_VideoColorBGRint_to_ABGRint(3MLIB) 1301
 mlib_VideoColorBlendABGR(3MLIB) 1303
 mlib_VideoColorBlendABGR_Inp(3MLIB) 1305
 mlib_VideoColorCMYK2JFIFYCCK444(3MLIB) 1307
 mlib_VideoColorJFIFYCC2ABGR444(3MLIB) 1308
 mlib_VideoColorJFIFYCC2ARGB444(3MLIB) 1309
 mlib_VideoColorJFIFYCC2RGB420(3MLIB) 1310
 mlib_VideoColorJFIFYCC2RGB420_Nearest(3MLIB) 1312
 mlib_VideoColorJFIFYCC2RGB422(3MLIB) 1313
 mlib_VideoColorJFIFYCC2RGB422_Nearest(3MLIB) 1314
 mlib_VideoColorJFIFYCC2RGB444(3MLIB) 1315
 mlib_VideoColorJFIFYCC2RGB444_S16(3MLIB) 1316
 mlib_VideoColorJFIFYCCK2CMYK444(3MLIB) 1317
 mlib_VideoColorMerge2(3MLIB) 1318
 mlib_VideoColorMerge2_S16(3MLIB) 1319
 mlib_VideoColorMerge3(3MLIB) 1320
 mlib_VideoColorMerge3_S16(3MLIB) 1321
 mlib_VideoColorMerge4(3MLIB) 1322
 mlib_VideoColorMerge4_S16(3MLIB) 1323
 mlib_VideoColorResizeABGR(3MLIB) 1324
 mlib_VideoColorRGB2ABGR(3MLIB) 1325
 mlib_VideoColorRGB2ARGB(3MLIB) 1326
 mlib_VideoColorRGB2JFIFYCC420(3MLIB) 1327
 mlib_VideoColorRGB2JFIFYCC422(3MLIB) 1328
 mlib_VideoColorRGB2JFIFYCC444(3MLIB) 1329
 mlib_VideoColorRGB2JFIFYCC444_S16(3MLIB) 1330
 mlib_VideoColorRGBAint_to_ABGRint(3MLIB) 1331
 mlib_VideoColorRGBint_to_ABGRint(3MLIB) 1332
 mlib_VideoColorRGBseq_to_ABGRint(3MLIB) 1334
 mlib_VideoColorRGBXint_to_ABGRint(3MLIB) 1336
 mlib_VideoColorRGBXint_to_ARGBint(3MLIB) 1338
 mlib_VideoColorSplit2(3MLIB) 1339
 mlib_VideoColorSplit2_S16(3MLIB) 1340
 mlib_VideoColorSplit3(3MLIB) 1341
 mlib_VideoColorSplit3_S16(3MLIB) 1342
 mlib_VideoColorSplit4(3MLIB) 1343

mllib_VideoColorSplit4_S16(3MLIB)	1344
mllib_VideoColorUYV444int_to_ABGRint(3MLIB)	1345
mllib_VideoColorUYV444int_to_ARGBint(3MLIB)	1347
mllib_VideoColorUYV444int_to_UYVY422int(3MLIB)	1349
mllib_VideoColorUYV444int_to_YUYV422int(3MLIB)	1350
mllib_VideoColorUYVY422int_to_ABGRint(3MLIB)	1351
mllib_VideoColorUYVY422int_to_ARGBint(3MLIB)	1353
mllib_VideoColorXRGBint_to_ABGRint(3MLIB)	1355
mllib_VideoColorXRGBint_to_ARGBint(3MLIB)	1356
mllib_VideoColorYUV2ABGR411(3MLIB)	1357
mllib_VideoColorYUV2ABGR420(3MLIB)	1359
mllib_VideoColorYUV2ABGR420_W(3MLIB)	1361
mllib_VideoColorYUV2ABGR420_WX2(3MLIB)	1363
mllib_VideoColorYUV2ABGR420_WX3(3MLIB)	1365
mllib_VideoColorYUV2ABGR420_X2(3MLIB)	1367
mllib_VideoColorYUV2ABGR420_X3(3MLIB)	1369
mllib_VideoColorYUV2ABGR422(3MLIB)	1371
mllib_VideoColorYUV2ABGR444(3MLIB)	1373
mllib_VideoColorYUV2ARGB411(3MLIB)	1375
mllib_VideoColorYUV2ARGB420(3MLIB)	1377
mllib_VideoColorYUV2ARGB422(3MLIB)	1379
mllib_VideoColorYUV2ARGB444(3MLIB)	1381
mllib_VideoColorYUV2RGB411(3MLIB)	1383
mllib_VideoColorYUV2RGB420(3MLIB)	1385
mllib_VideoColorYUV2RGB422(3MLIB)	1387
mllib_VideoColorYUV2RGB444(3MLIB)	1389
mllib_VideoColorYUV411seq_to_ABGRint(3MLIB)	1391
mllib_VideoColorYUV411seq_to_ARGBint(3MLIB)	1393
mllib_VideoColorYUV411seq_to_UYVY422int(3MLIB)	1395
mllib_VideoColorYUV411seq_to_YUYV422int(3MLIB)	1397
mllib_VideoColorYUV420seq_to_ABGRint(3MLIB)	1399
mllib_VideoColorYUV420seq_to_ARGBint(3MLIB)	1401
mllib_VideoColorYUV420seq_to_UYVY422int(3MLIB)	1403
mllib_VideoColorYUV420seq_to_YUYV422int(3MLIB)	1405
mllib_VideoColorYUV422seq_to_ABGRint(3MLIB)	1407
mllib_VideoColorYUV422seq_to_ARGBint(3MLIB)	1409
mllib_VideoColorYUV422seq_to_UYVY422int(3MLIB)	1411
mllib_VideoColorYUV422seq_to_YUYV422int(3MLIB)	1413

mlib_VideoColorYUV444int_to_ABGRint(3MLIB) 1415
 mlib_VideoColorYUV444int_to_ARGBint(3MLIB) 1417
 mlib_VideoColorYUV444int_to_UYVY422int(3MLIB) 1419
 mlib_VideoColorYUV444int_to_YUYV422int(3MLIB) 1420
 mlib_VideoColorYUV444seq_to_ABGRint(3MLIB) 1421
 mlib_VideoColorYUV444seq_to_ARGBint(3MLIB) 1423
 mlib_VideoColorYUV444seq_to_UYVY422int(3MLIB) 1425
 mlib_VideoColorYUV444seq_to_YUYV422int(3MLIB) 1426
 mlib_VideoColorYUYV422int_to_ABGRint(3MLIB) 1427
 mlib_VideoColorYUYV422int_to_ARGBint(3MLIB) 1429
 mlib_VideoCopyRefAve_U8_U8_16x16(3MLIB) 1431
 mlib_VideoCopyRefAve_U8_U8(3MLIB) 1433
 mlib_VideoCopyRef_S16_U8_16x16(3MLIB) 1435
 mlib_VideoCopyRef_S16_U8(3MLIB) 1437
 mlib_VideoCopyRef_U8_U8_16x16(3MLIB) 1439
 mlib_VideoCopyRef_U8_U8(3MLIB) 1441
 mlib_VideoDCT16x16_S16_S16(3MLIB) 1443
 mlib_VideoDCT16x16_S16_S16_B10(3MLIB) 1444
 mlib_VideoDCT2x2_S16_S16(3MLIB) 1445
 mlib_VideoDCT4x4_S16_S16(3MLIB) 1446
 mlib_VideoDCT8x8_S16_S16(3MLIB) 1447
 mlib_VideoDCT8x8_S16_S16_B12(3MLIB) 1448
 mlib_VideoDCT8x8_S16_S16_NA(3MLIB) 1449
 mlib_VideoDCT8x8_S16_U8(3MLIB) 1450
 mlib_VideoDCT8x8_S16_U8_NA(3MLIB) 1451
 mlib_VideoDeQuantizeInit_S16(3MLIB) 1452
 mlib_VideoDeQuantize_S16(3MLIB) 1453
 mlib_VideoDownSample420(3MLIB) 1454
 mlib_VideoDownSample420_S16(3MLIB) 1455
 mlib_VideoDownSample422(3MLIB) 1456
 mlib_VideoDownSample422_S16(3MLIB) 1457
 mlib_VideoH263OverlappedMC_S16_U8(3MLIB) 1458
 mlib_VideoH263OverlappedMC_U8_U8(3MLIB) 1461
 mlib_VideoIDCT8x8_S16_S16(3MLIB) 1464
 mlib_VideoIDCT8x8_S16_S16_DC(3MLIB) 1465
 mlib_VideoIDCT8x8_S16_S16_NA(3MLIB) 1466
 mlib_VideoIDCT8x8_S16_S16_Q1(3MLIB) 1467
 mlib_VideoIDCT8x8_S16_S16_Q1_Mismatch(3MLIB) 1468

mlib_VideoIDCT8x8_U8_S16(3MLIB) 1469
 mlib_VideoIDCT8x8_U8_S16_DC(3MLIB) 1470
 mlib_VideoIDCT8x8_U8_S16_NA(3MLIB) 1471
 mlib_VideoIDCT8x8_U8_S16_Q1(3MLIB) 1472
 mlib_VideoIDCT_IEEE_S16_S16(3MLIB) 1473
 mlib_VideoInterpAveX_U8_U8_16x16(3MLIB) 1474
 mlib_VideoInterpAveX_U8_U8(3MLIB) 1476
 mlib_VideoInterpAveXY_U8_U8_16x16(3MLIB) 1478
 mlib_VideoInterpAveXY_U8_U8(3MLIB) 1480
 mlib_VideoInterpAveY_U8_U8_16x16(3MLIB) 1482
 mlib_VideoInterpAveY_U8_U8(3MLIB) 1484
 mlib_VideoInterpX_S16_U8_16x16(3MLIB) 1486
 mlib_VideoInterpX_S16_U8(3MLIB) 1488
 mlib_VideoInterpX_U8_U8_16x16(3MLIB) 1490
 mlib_VideoInterpX_U8_U8(3MLIB) 1492
 mlib_VideoInterpXY_S16_U8_16x16(3MLIB) 1494
 mlib_VideoInterpXY_S16_U8(3MLIB) 1496
 mlib_VideoInterpXY_U8_U8_16x16(3MLIB) 1498
 mlib_VideoInterpXY_U8_U8(3MLIB) 1500
 mlib_VideoInterpX_Y_XY_U8_U8(3MLIB) 1502
 mlib_VideoInterpY_S16_U8_16x16(3MLIB) 1503
 mlib_VideoInterpY_S16_U8(3MLIB) 1505
 mlib_VideoInterpY_U8_U8_16x16(3MLIB) 1507
 mlib_VideoInterpY_U8_U8(3MLIB) 1509
 mlib_VideoP64Decimate_U8_U8(3MLIB) 1511
 mlib_VideoP64Loop_S16_U8(3MLIB) 1513
 mlib_VideoP64Loop_U8_U8(3MLIB) 1515
 mlib_VideoQuantizeInit_S16(3MLIB) 1517
 mlib_VideoQuantize_S16(3MLIB) 1518
 mlib_VideoReversibleColorRGB2YUV_U8_U8(3MLIB) 1519
 mlib_VideoReversibleColorYUV2RGB_U8_U8(3MLIB) 1521
 mlib_VideoSignMagnitudeConvert_S16(3MLIB) 1523
 mlib_VideoSignMagnitudeConvert_S16_S16(3MLIB) 1524
 mlib_VideoSignMagnitudeConvert_S32(3MLIB) 1525
 mlib_VideoSignMagnitudeConvert_S32_S32(3MLIB) 1526
 mlib_VideoSumAbsDiff(3MLIB) 1527
 mlib_VideoUpSample420(3MLIB) 1528
 mlib_VideoUpSample420_Nearest(3MLIB) 1529

mllib_VideoUpSample420_Nearest_S16(3MLIB)	1530
mllib_VideoUpSample420_S16(3MLIB)	1531
mllib_VideoUpSample422(3MLIB)	1532
mllib_VideoUpSample422_Nearest(3MLIB)	1533
mllib_VideoUpSample422_Nearest_S16(3MLIB)	1534
mllib_VideoUpSample422_S16(3MLIB)	1535
mllib_VideoWaveletForwardTwoTenTrans(3MLIB)	1536
mllib_VideoWaveletInverseTwoTenTrans(3MLIB)	1537
mllib_VolumeFindMaxBMask_U8(3MLIB)	1538
mllib_VolumeFindMaxCMask_U8(3MLIB)	1539
mllib_VolumeFindMax_U8(3MLIB)	1540
mllib_VolumeRayCast_Blocked(3MLIB)	1541
mllib_VolumeRayCast_General(3MLIB)	1543
mllib_VolumeWindowLevel(3MLIB)	1545

Index	1547
--------------	-------------

Preface

Both novice users and those familiar with the SunOS operating system can use online man pages to obtain information about the system and its features. A man page is intended to answer concisely the question “What does it do?” The man pages in general comprise a reference manual. They are not intended to be a tutorial.

Overview

The following contains a brief description of each man page section and the information it references:

- Section 1 describes, in alphabetical order, commands available with the operating system.
- Section 1M describes, in alphabetical order, commands that are used chiefly for system maintenance and administration purposes.
- Section 2 describes all of the system calls. Most of these calls have one or more error returns. An error condition is indicated by an otherwise impossible returned value.
- Section 3 describes functions found in various libraries, other than those functions that directly invoke UNIX system primitives, which are described in Section 2.
- Section 4 outlines the formats of various files. The C structure declarations for the file formats are given where applicable.
- Section 5 contains miscellaneous documentation such as character-set tables.
- Section 6 contains available games and demos.
- Section 7 describes various special files that refer to specific hardware peripherals and device drivers. STREAMS software drivers, modules and the STREAMS-generic set of system calls are also described.

- Section 9 provides reference information needed to write device drivers in the kernel environment. It describes two device driver interface specifications: the Device Driver Interface (DDI) and the Driver/Kernel Interface (DKI).
- Section 9E describes the DDI/DKI, DDI-only, and DKI-only entry-point routines a developer can include in a device driver.
- Section 9F describes the kernel functions available for use by device drivers.
- Section 9S describes the data structures used by drivers to share information between the driver and the kernel.

Below is a generic format for man pages. The man pages of each manual section generally follow this order, but include only needed headings. For example, if there are no bugs to report, there is no BUGS section. See the `intro` pages for more information and detail about each section, and `man(1)` for more information about man pages in general.

NAME	This section gives the names of the commands or functions documented, followed by a brief description of what they do.								
SYNOPSIS	<p>This section shows the syntax of commands or functions. When a command or file does not exist in the standard path, its full path name is shown. Options and arguments are alphabetized, with single letter arguments first, and options with arguments next, unless a different argument order is required.</p> <p>The following special characters are used in this section:</p> <table border="0" style="margin-left: 20px;"> <tr> <td style="padding-right: 10px;">[]</td> <td>Brackets. The option or argument enclosed in these brackets is optional. If the brackets are omitted, the argument must be specified.</td> </tr> <tr> <td style="padding-right: 10px;">. . .</td> <td>Ellipses. Several values can be provided for the previous argument, or the previous argument can be specified multiple times, for example, "filename . . .".</td> </tr> <tr> <td style="padding-right: 10px;"> </td> <td>Separator. Only one of the arguments separated by this character can be specified at a time.</td> </tr> <tr> <td style="padding-right: 10px;">{ }</td> <td>Braces. The options and/or arguments enclosed within braces are interdependent, such that everything enclosed must be treated as a unit.</td> </tr> </table>	[]	Brackets. The option or argument enclosed in these brackets is optional. If the brackets are omitted, the argument must be specified.	. . .	Ellipses. Several values can be provided for the previous argument, or the previous argument can be specified multiple times, for example, "filename . . .".		Separator. Only one of the arguments separated by this character can be specified at a time.	{ }	Braces. The options and/or arguments enclosed within braces are interdependent, such that everything enclosed must be treated as a unit.
[]	Brackets. The option or argument enclosed in these brackets is optional. If the brackets are omitted, the argument must be specified.								
. . .	Ellipses. Several values can be provided for the previous argument, or the previous argument can be specified multiple times, for example, "filename . . .".								
	Separator. Only one of the arguments separated by this character can be specified at a time.								
{ }	Braces. The options and/or arguments enclosed within braces are interdependent, such that everything enclosed must be treated as a unit.								

PROTOCOL	This section occurs only in subsection 3R to indicate the protocol description file.
DESCRIPTION	This section defines the functionality and behavior of the service. Thus it describes concisely what the command does. It does not discuss OPTIONS or cite EXAMPLES. Interactive commands, subcommands, requests, macros, and functions are described under USAGE.
IOCTL	This section appears on pages in Section 7 only. Only the device class that supplies appropriate parameters to the <code>ioctl(2)</code> system call is called <code>ioctl</code> and generates its own heading. <code>ioctl</code> calls for a specific device are listed alphabetically (on the man page for that specific device). <code>ioctl</code> calls are used for a particular class of devices all of which have an <code>io</code> ending, such as <code>mtio(7I)</code> .
OPTIONS	This section lists the command options with a concise summary of what each option does. The options are listed literally and in the order they appear in the SYNOPSIS section. Possible arguments to options are discussed under the option, and where appropriate, default values are supplied.
OPERANDS	This section lists the command operands and describes how they affect the actions of the command.
OUTPUT	This section describes the output – standard output, standard error, or output files – generated by the command.
RETURN VALUES	If the man page documents functions that return values, this section lists these values and describes the conditions under which they are returned. If a function can return only constant values, such as 0 or -1, these values are listed in tagged paragraphs. Otherwise, a single paragraph describes the return values of each function. Functions declared void do not return values, so they are not discussed in RETURN VALUES.
ERRORS	On failure, most functions place an error code in the global variable <code>errno</code> indicating why they failed. This section lists alphabetically all error codes a function can generate and describes the

	<p>conditions that cause each error. When more than one condition can cause the same error, each condition is described in a separate paragraph under the error code.</p>
USAGE	<p>This section lists special rules, features, and commands that require in-depth explanations. The subsections listed here are used to explain built-in functionality:</p> <ul style="list-style-type: none"> Commands Modifiers Variables Expressions Input Grammar
EXAMPLES	<p>This section provides examples of usage or of how to use a command or function. Wherever possible a complete example including command-line entry and machine response is shown. Whenever an example is given, the prompt is shown as <code>example%</code>, or if the user must be superuser, <code>example#</code>. Examples are followed by explanations, variable substitution rules, or returned values. Most examples illustrate concepts from the SYNOPSIS, DESCRIPTION, OPTIONS, and USAGE sections.</p>
ENVIRONMENT VARIABLES	<p>This section lists any environment variables that the command or function affects, followed by a brief description of the effect.</p>
EXIT STATUS	<p>This section lists the values the command returns to the calling program or shell and the conditions that cause these values to be returned. Usually, zero is returned for successful completion, and values other than zero for various error conditions.</p>
FILES	<p>This section lists all file names referred to by the man page, files of interest, and files created or required by commands. Each is followed by a descriptive summary or explanation.</p>
ATTRIBUTES	<p>This section lists characteristics of commands, utilities, and device drivers by defining the attribute type and its corresponding value. See <code>attributes(5)</code> for more information.</p>
SEE ALSO	<p>This section lists references to other man pages, in-house documentation, and outside publications.</p>

DIAGNOSTICS	This section lists diagnostic messages with a brief explanation of the condition causing the error.
WARNINGS	This section lists warnings about special conditions which could seriously affect your working conditions. This is not a list of diagnostics.
NOTES	This section lists additional information that does not belong anywhere else on the page. It takes the form of an aside to the user, covering points of special interest. Critical information is never covered here.
BUGS	This section describes known bugs and, wherever possible, suggests workarounds.

Multimedia Library Functions Part One

mllib_free(3MLIB)

NAME mllib_free – free a block of bytes

SYNOPSIS cc [*flag...*] *file...* -lmllib [*library...*]
#include <mllib.h>

```
void mllib_free(void *ptr);
```

DESCRIPTION The mllib_free() function frees a block of bytes previously allocated by mllib_malloc() or mllib_realloc().

This function is a wrapper of the standard C function free().

PARAMETERS The function takes the following arguments:

ptr Pointer to a previously allocated block.

RETURN VALUES None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_malloc(3MLIB), mllib_realloc(3MLIB), malloc(3C), attributes(5)

mllib_GraphicsBoundaryFill_8(3MLIB)

NAME mllib_GraphicsBoundaryFill_8, mllib_GraphicsBoundaryFill_32 – boundary fill

SYNOPSIS cc [*flag...*] *file...* -lmllib [*library...*]
 #include <mllib.h>

```
mllib_status mllib_GraphicsBoundaryFill_8(mllib_image *buffer,
    mllib_s16 x, mllib_s16 y, mllib_s32 c, mllib_s32 c2);
mllib_status mllib_GraphicsBoundaryFill_32(mllib_image *buffer,
    mllib_s16 x, mllib_s16 y, mllib_s32 c, mllib_s32 c2);
```

DESCRIPTION Each of these functions performs boundary fill.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

x X coordinate of the starting point.

y Y coordinate of the starting point.

c Color used in the drawing.

c2 Color that defines the filling boundary.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO attributes(5)

mllib_GraphicsDrawArc_8(3MLIB)

NAME	mllib_GraphicsDrawArc_8, mllib_GraphicsDrawArc_32 – draw arc														
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_GraphicsDrawArc_8(mllib_image *buffer, mllib_s16 x, mllib_s16 y, mllib_s32 r, mllib_f32 t1, mllib_f32 t2, mllib_s32 c); mllib_status mllib_GraphicsDrawArc_32(mllib_image *buffer, mllib_s16 x, mllib_s16 y, mllib_s32 r, mllib_f32 t1, mllib_f32 t2, mllib_s32 c);</pre>														
DESCRIPTION	Each of these functions draws an arc with the center at (x,y), radius r, start angle q1, and end angle q2.														
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>x</i></td><td>X coordinate of the center.</td></tr><tr><td><i>y</i></td><td>Y coordinate of the center.</td></tr><tr><td><i>r</i></td><td>Radius of the arc.</td></tr><tr><td><i>t1</i></td><td>Start angle of the arc in radians.</td></tr><tr><td><i>t2</i></td><td>End angle of the arc in radians.</td></tr><tr><td><i>c</i></td><td>Color used in the drawing.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>x</i>	X coordinate of the center.	<i>y</i>	Y coordinate of the center.	<i>r</i>	Radius of the arc.	<i>t1</i>	Start angle of the arc in radians.	<i>t2</i>	End angle of the arc in radians.	<i>c</i>	Color used in the drawing.
<i>buffer</i>	Pointer to the image into which the function is drawing.														
<i>x</i>	X coordinate of the center.														
<i>y</i>	Y coordinate of the center.														
<i>r</i>	Radius of the arc.														
<i>t1</i>	Start angle of the arc in radians.														
<i>t2</i>	End angle of the arc in radians.														
<i>c</i>	Color used in the drawing.														
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.														
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe								
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														
MT-Level	MT-Safe														
SEE ALSO	mllib_GraphicsDrawArc_A_8(3MLIB), mllib_GraphicsDrawArc_X_8(3MLIB), attributes(5)														

NAME mllib_GraphicsDrawArc_A_8, mllib_GraphicsDrawArc_A_32 – draw arc with antialiasing

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawArc_A_8(mllib_image *buffer, mllib_s16
    x, mllib_s16 y, mllib_s32 r, mllib_f32 t1, mllib_f32 t2, mllib_s32
    c);

mllib_status mllib_GraphicsDrawArc_A_32(mllib_image *buffer, mllib_s16
    x, mllib_s16 y, mllib_s32 r, mllib_f32 t1, mllib_f32 t2, mllib_s32
    c);
```

DESCRIPTION Each of these functions draws an arc with the center at (x,y), radius r, start angle q1, and end angle q2.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

x X coordinate of the center.

y Y coordinate of the center.

r Radius of the arc.

t1 Start angle of the arc in radians.

t2 End angle of the arc in radians.

c Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawArc_8(3MLIB), mllib_GraphicsDrawArc_X_8(3MLIB), attributes(5)

mllib_GraphicsDrawArc_X_8(3MLIB)

NAME	mllib_GraphicsDrawArc_X_8, mllib_GraphicsDrawArc_X_32 – draw arc in Xor mode																
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_GraphicsDrawArc_X_8(mllib_image *buffer, mllib_s16 x, mllib_s16 y, mllib_s32 r, mllib_f32 t1, mllib_f32 t2, mllib_s32 c, mllib_s32 c2); mllib_status mllib_GraphicsDrawArc_X_32(mllib_image *buffer, mllib_s16 x, mllib_s16 y, mllib_s32 r, mllib_f32 t1, mllib_f32 t2, mllib_s32 c, mllib_s32 c2);</pre>																
DESCRIPTION	Each of these functions draws an arc with the center at (x,y), radius r, start angle q1, and end angle q2.																
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>x</i></td><td>X coordinate of the center.</td></tr><tr><td><i>y</i></td><td>Y coordinate of the center.</td></tr><tr><td><i>r</i></td><td>Radius of the arc.</td></tr><tr><td><i>t1</i></td><td>Start angle of the arc in radians.</td></tr><tr><td><i>t2</i></td><td>End angle of the arc in radians.</td></tr><tr><td><i>c</i></td><td>Color used in the drawing.</td></tr><tr><td><i>c2</i></td><td>Alternation color.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>x</i>	X coordinate of the center.	<i>y</i>	Y coordinate of the center.	<i>r</i>	Radius of the arc.	<i>t1</i>	Start angle of the arc in radians.	<i>t2</i>	End angle of the arc in radians.	<i>c</i>	Color used in the drawing.	<i>c2</i>	Alternation color.
<i>buffer</i>	Pointer to the image into which the function is drawing.																
<i>x</i>	X coordinate of the center.																
<i>y</i>	Y coordinate of the center.																
<i>r</i>	Radius of the arc.																
<i>t1</i>	Start angle of the arc in radians.																
<i>t2</i>	End angle of the arc in radians.																
<i>c</i>	Color used in the drawing.																
<i>c2</i>	Alternation color.																
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.																
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe										
ATTRIBUTE TYPE	ATTRIBUTE VALUE																
Interface Stability	Evolving																
MT-Level	MT-Safe																
SEE ALSO	mllib_GraphicsDrawArc_8(3MLIB), mllib_GraphicsDrawArc_A_8(3MLIB), attributes(5)																

mllib_GraphicsDrawCircle_8(3MLIB)

NAME mllib_GraphicsDrawCircle_8, mllib_GraphicsDrawCircle_32 – draw circle

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawCircle_8(mllib_image *buffer, mllib_s16
    x, mllib_s16 y, mllib_s32 r, mllib_s32 c);

mllib_status mllib_GraphicsDrawCircle_32(mllib_image *buffer, mllib_s16
    x, mllib_s16 y, mllib_s32 r, mllib_s32 c);
```

DESCRIPTION Each of these functions draws a circle with the center at (x,y) and radius r.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

x X coordinate of the center.

y Y coordinate of the center.

r Radius of the arc.

c Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawCircle_A_8(3MLIB),
mllib_GraphicsDrawCircle_X_8(3MLIB), attributes(5)

mllib_GraphicsDrawCircle_A_8(3MLIB)

NAME mllib_GraphicsDrawCircle_A_8, mllib_GraphicsDrawCircle_A_32 – draw circle with antialiasing

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawCircle_A_8(mllib_image *buffer,
      mllib_s16 x, mllib_s16 y, mllib_s32 r, mllib_s32 c);

mllib_status mllib_GraphicsDrawCircle_A_32(mllib_image *buffer,
      mllib_s16 x, mllib_s16 y, mllib_s32 r, mllib_s32 c);
```

DESCRIPTION Each of these functions draws a circle with the center at (x,y) and radius r.

PARAMETERS Each of the functions takes the following arguments:

<i>buffer</i>	Pointer to the image into which the function is drawing.
<i>x</i>	X coordinate of the center.
<i>y</i>	Y coordinate of the center.
<i>r</i>	Radius of the arc.
<i>c</i>	Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawCircle_8(3MLIB), mllib_GraphicsDrawCircle_X_8(3MLIB), attributes(5)

mllib_GraphicsDrawCircle_X_8(3MLIB)

NAME	mllib_GraphicsDrawCircle_X_8, mllib_GraphicsDrawCircle_X_32 – draw circle in Xor mode						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawCircle_X_8(mllib_image *<i>buffer</i>, mllib_s16 <i>x</i>, mllib_s16 <i>y</i>, mllib_s32 <i>r</i>, mllib_s32 <i>c</i>, mllib_s32 <i>c2</i>); mllib_status mllib_GraphicsDrawCircle_X_32(mllib_image *<i>buffer</i>, mllib_s16 <i>x</i>, mllib_s16 <i>y</i>, mllib_s32 <i>r</i>, mllib_s32 <i>c</i>, mllib_s32 <i>c2</i>);</pre>						
DESCRIPTION	Each of these functions draws a circle with the center at (x,y) and radius r.						
PARAMETERS	Each of the functions takes the following arguments: <i>buffer</i> Pointer to the image into which the function is drawing. <i>x</i> X coordinate of the center. <i>y</i> Y coordinate of the center. <i>r</i> Radius of the arc. <i>c</i> Color used in the drawing. <i>c2</i> Alternation color.						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_GraphicsDrawCircle_8(3MLIB), mllib_GraphicsDrawCircle_A_8(3MLIB), attributes(5)						

mllib_GraphicsDrawEllipse_8(3MLIB)

NAME	mllib_GraphicsDrawEllipse_8, mllib_GraphicsDrawEllipse_32 – draw ellipse														
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawEllipse_8(mllib_image *<i>buffer</i>, mllib_s16 <i>x</i>, mllib_s16 <i>y</i>, mllib_s32 <i>a</i>, mllib_s32 <i>b</i>, mllib_f32 <i>t</i>, mllib_s32 <i>c</i>); mllib_status mllib_GraphicsDrawEllipse_32(mllib_image *<i>buffer</i>, mllib_s16 <i>x</i>, mllib_s16 <i>y</i>, mllib_s32 <i>a</i>, mllib_s32 <i>b</i>, mllib_f32 <i>t</i>, mllib_s32 <i>c</i>);</pre>														
DESCRIPTION	Each of these functions draws an ellipse with the center at (x, y) , major semiaxis a , and minor semiaxis b . The angle of the major semiaxis is t counterclockwise from the X axis.														
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>x</i></td><td>X coordinate of the center.</td></tr><tr><td><i>y</i></td><td>Y coordinate of the center.</td></tr><tr><td><i>a</i></td><td>Major semiaxis of the ellipse.</td></tr><tr><td><i>b</i></td><td>Minor semiaxis of the ellipse.</td></tr><tr><td><i>t</i></td><td>Angle of major semiaxis in radians.</td></tr><tr><td><i>c</i></td><td>Color used in the drawing.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>x</i>	X coordinate of the center.	<i>y</i>	Y coordinate of the center.	<i>a</i>	Major semiaxis of the ellipse.	<i>b</i>	Minor semiaxis of the ellipse.	<i>t</i>	Angle of major semiaxis in radians.	<i>c</i>	Color used in the drawing.
<i>buffer</i>	Pointer to the image into which the function is drawing.														
<i>x</i>	X coordinate of the center.														
<i>y</i>	Y coordinate of the center.														
<i>a</i>	Major semiaxis of the ellipse.														
<i>b</i>	Minor semiaxis of the ellipse.														
<i>t</i>	Angle of major semiaxis in radians.														
<i>c</i>	Color used in the drawing.														
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.														
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe								
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														
MT-Level	MT-Safe														
SEE ALSO	mllib_GraphicsDrawEllipse_A_8(3MLIB) , mllib_GraphicsDrawEllipse_X_8(3MLIB) , attributes(5)														

mllib_GraphicsDrawEllipse_A_8(3MLIB)

NAME mllib_GraphicsDrawEllipse_A_8, mllib_GraphicsDrawEllipse_A_32 – draw ellipse with antialiasing

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawEllipse_A_8(mllib_image *buffer,
      mllib_s16 x, mllib_s16 y, mllib_s32 a, mllib_s32 b, mllib_f32 t,
      mllib_s32 c);

mllib_status mllib_GraphicsDrawEllipse_A_32(mllib_image *buffer,
      mllib_s16 x, mllib_s16 y, mllib_s32 a, mllib_s32 b, mllib_f32 t,
      mllib_s32 c);
```

DESCRIPTION Each of these functions draws an ellipse with the center at (*x*, *y*), major semiaxis *a*, and minor semiaxis *b*. The angle of the major semiaxis is *t* counterclockwise from the X axis.

PARAMETERS Each of the functions takes the following arguments:

<i>buffer</i>	Pointer to the image into which the function is drawing.
<i>x</i>	X coordinate of the center.
<i>y</i>	Y coordinate of the center.
<i>a</i>	Major semiaxis of the ellipse.
<i>b</i>	Minor semiaxis of the ellipse.
<i>t</i>	Angle of major semiaxis in radians.
<i>c</i>	Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_GraphicsDrawEllipse_8\(3MLIB\)](#), [mllib_GraphicsDrawEllipse_X_8\(3MLIB\)](#), [attributes\(5\)](#)

mllib_GraphicsDrawEllipse_X_8(3MLIB)

NAME	mllib_GraphicsDrawEllipse_X_8, mllib_GraphicsDrawEllipse_X_32 – draw ellipse in Xor mode						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawEllipse_X_8(mllib_image *<i>buffer</i>, mllib_s16 <i>x</i>, mllib_s16 <i>y</i>, mllib_s32 <i>a</i>, mllib_s32 <i>b</i>, mllib_f32 <i>t</i>, mllib_s32 <i>c</i>, mllib_s32 <i>c2</i>); mllib_status mllib_GraphicsDrawEllipse_X_32(mllib_image *<i>buffer</i>, mllib_s16 <i>x</i>, mllib_s16 <i>y</i>, mllib_s32 <i>a</i>, mllib_s32 <i>b</i>, mllib_f32 <i>t</i>, mllib_s32 <i>c</i>, mllib_s32 <i>c2</i>);</pre>						
DESCRIPTION	Each of these functions draws an ellipse with the center at (<i>x</i> , <i>y</i>), major semiaxis <i>a</i> , and minor semiaxis <i>b</i> . The angle of the major semiaxis is <i>t</i> counterclockwise from the X axis.						
PARAMETERS	Each of the functions takes the following arguments: <i>buffer</i> Pointer to the image into which the function is drawing. <i>x</i> X coordinate of the center. <i>y</i> Y coordinate of the center. <i>a</i> Major semiaxis of the ellipse. <i>b</i> Minor semiaxis of the ellipse. <i>t</i> Angle of major semiaxis in radians. <i>c</i> Color used in the drawing. <i>c2</i> Alternation color.						
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_GraphicsDrawEllipse_8(3MLIB), mllib_GraphicsDrawEllipse_A_8(3MLIB), attributes(5)						

mllib_GraphicsDrawLine_8(3MLIB)

NAME mllib_GraphicsDrawLine_8, mllib_GraphicsDrawLine_32 – draw line

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawLine_8(mllib_image *buffer, mllib_s16
    x1, mllib_s16 y1, mllib_s16 x2, mllib_s16 y2, mllib_s32 c);

mllib_status mllib_GraphicsDrawLine_32(mllib_image *buffer, mllib_s16
    x1, mllib_s16 y1, mllib_s16 x2, mllib_s16 y2, mllib_s32 c);
```

DESCRIPTION Each of these functions draws a line between (x1,y1) and (x2,y2).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.
x1 X coordinate of the first point.
y1 Y coordinate of the first point.
x2 X coordinate of the second point.
y2 Y coordinate of the second point.
c Color used in the drawing.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawLine_A_8(3MLIB),
 mllib_GraphicsDrawLine_AG_8(3MLIB),
 mllib_GraphicsDrawLine_AGZ_8(3MLIB),
 mllib_GraphicsDrawLine_AZ_8(3MLIB),
 mllib_GraphicsDrawLine_G_8(3MLIB),
 mllib_GraphicsDrawLine_GZ_8(3MLIB),
 mllib_GraphicsDrawLine_X_8(3MLIB), mllib_GraphicsDrawLine_Z_8(3MLIB),
 attributes(5)

mllib_GraphicsDrawLine_A_8(3MLIB)

NAME mllib_GraphicsDrawLine_A_8, mllib_GraphicsDrawLine_A_32 – draw line with antialiasing

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawLine_A_8(mllib_image *buffer, mllib_s16
    x1, mllib_s16 y1, mllib_s16 x2, mllib_s16 y2, mllib_s32 c);

mllib_status mllib_GraphicsDrawLine_A_32(mllib_image *buffer, mllib_s16
    x1, mllib_s16 y1, mllib_s16 x2, mllib_s16 y2, mllib_s32 c);
```

DESCRIPTION Each of these functions draws a line between (x1,y1) and (x2,y2).

PARAMETERS Each of the functions takes the following arguments:

<i>buffer</i>	Pointer to the image into which the function is drawing.
<i>x1</i>	X coordinate of the first point.
<i>y1</i>	Y coordinate of the first point.
<i>x2</i>	X coordinate of the second point.
<i>y2</i>	Y coordinate of the second point.
<i>c</i>	Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawLine_8(3MLIB), mllib_GraphicsDrawLine_AG_8(3MLIB), mllib_GraphicsDrawLine_AGZ_8(3MLIB), mllib_GraphicsDrawLine_AZ_8(3MLIB), mllib_GraphicsDrawLine_G_8(3MLIB), mllib_GraphicsDrawLine_GZ_8(3MLIB), mllib_GraphicsDrawLine_X_8(3MLIB), mllib_GraphicsDrawLine_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawLine_AG_8(3MLIB)

NAME mllib_GraphicsDrawLine_AG_8, mllib_GraphicsDrawLine_AG_32 – draw line with antialiasing and Gouraud shading

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawLine_AG_8(mllib_image *buffer, mllib_s16
    x1, mllib_s16 y1, mllib_s16 x2, mllib_s16 y2, mllib_s32 c1,
    mllib_s32 c2);

mllib_status mllib_GraphicsDrawLine_AG_32(mllib_image *buffer,
    mllib_s16 x1, mllib_s16 y1, mllib_s16 x2, mllib_s16 y2, mllib_s32
    c1, mllib_s32 c2);
```

DESCRIPTION Each of these functions draws a line between (x1,y1) and (x2,y2).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

x1 X coordinate of the first point.

y1 Y coordinate of the first point.

x2 X coordinate of the second point.

y2 Y coordinate of the second point.

c1 Color of the first point.

c2 Color of the second point.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawLine_8(3MLIB), mllib_GraphicsDrawLine_A_8(3MLIB), mllib_GraphicsDrawLine_AGZ_8(3MLIB), mllib_GraphicsDrawLine_AZ_8(3MLIB), mllib_GraphicsDrawLine_G_8(3MLIB), mllib_GraphicsDrawLine_GZ_8(3MLIB), mllib_GraphicsDrawLine_X_8(3MLIB), mllib_GraphicsDrawLine_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawLine_AGZ_8(3MLIB)

NAME	mllib_GraphicsDrawLine_AGZ_8, mllib_GraphicsDrawLine_AGZ_32 – draw line with antialiasing, Gouraud shading, and Z buffering																				
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawLine_AGZ_8(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, mllib_s16 <i>x1</i>, mllib_s16 <i>y1</i>, mllib_s16 <i>z1</i>, mllib_s16 <i>x2</i>, mllib_s16 <i>y2</i>, mllib_s16 <i>z2</i>, mllib_s32 <i>c1</i>, mllib_s32 <i>c2</i>); mllib_status mllib_GraphicsDrawLine_AGZ_32(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, mllib_s16 <i>x1</i>, mllib_s16 <i>y1</i>, mllib_s16 <i>z1</i>, mllib_s16 <i>x2</i>, mllib_s16 <i>y2</i>, mllib_s16 <i>z2</i>, mllib_s32 <i>c1</i>, mllib_s32 <i>c2</i>);</pre>																				
DESCRIPTION	Each of these functions draws a between (x1,y1) and (x2,y2).																				
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>zbuffer</i></td><td>Pointer to the image that holds the Z buffer.</td></tr><tr><td><i>x1</i></td><td>X coordinate of the first point.</td></tr><tr><td><i>y1</i></td><td>Y coordinate of the first point.</td></tr><tr><td><i>z1</i></td><td>Z coordinate of the first point.</td></tr><tr><td><i>x2</i></td><td>X coordinate of the second point.</td></tr><tr><td><i>y2</i></td><td>Y coordinate of the second point.</td></tr><tr><td><i>z2</i></td><td>Z coordinate of the second point.</td></tr><tr><td><i>c1</i></td><td>Color of the first point.</td></tr><tr><td><i>c2</i></td><td>Color of the second point.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>zbuffer</i>	Pointer to the image that holds the Z buffer.	<i>x1</i>	X coordinate of the first point.	<i>y1</i>	Y coordinate of the first point.	<i>z1</i>	Z coordinate of the first point.	<i>x2</i>	X coordinate of the second point.	<i>y2</i>	Y coordinate of the second point.	<i>z2</i>	Z coordinate of the second point.	<i>c1</i>	Color of the first point.	<i>c2</i>	Color of the second point.
<i>buffer</i>	Pointer to the image into which the function is drawing.																				
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.																				
<i>x1</i>	X coordinate of the first point.																				
<i>y1</i>	Y coordinate of the first point.																				
<i>z1</i>	Z coordinate of the first point.																				
<i>x2</i>	X coordinate of the second point.																				
<i>y2</i>	Y coordinate of the second point.																				
<i>z2</i>	Z coordinate of the second point.																				
<i>c1</i>	Color of the first point.																				
<i>c2</i>	Color of the second point.																				
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.																				
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe														
ATTRIBUTE TYPE	ATTRIBUTE VALUE																				
Interface Stability	Evolving																				
MT-Level	MT-Safe																				

`mllib_GraphicsDrawLine_AGZ_8(3MLIB)`

SEE ALSO `mllib_GraphicsDrawLine_8(3MLIB)`, `mllib_GraphicsDrawLine_A_8(3MLIB)`,
`mllib_GraphicsDrawLine_AG_8(3MLIB)`,
`mllib_GraphicsDrawLine_AZ_8(3MLIB)`,
`mllib_GraphicsDrawLine_G_8(3MLIB)`,
`mllib_GraphicsDrawLine_GZ_8(3MLIB)`,
`mllib_GraphicsDrawLine_X_8(3MLIB)`, `mllib_GraphicsDrawLine_Z_8(3MLIB)`,
`attributes(5)`

mllib_GraphicsDrawLine_AZ_8(3MLIB)

NAME	mllib_GraphicsDrawLine_AZ_8, mllib_GraphicsDrawLine_AZ_32 – draw line with antialiasing and Z buffering																		
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawLine_AZ_8(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, mllib_s16 <i>x1</i>, mllib_s16 <i>y1</i>, mllib_s16 <i>z1</i>, mllib_s16 <i>x2</i>, mllib_s16 <i>y2</i>, mllib_s16 <i>z2</i>, mllib_s32 <i>c</i>); mllib_status mllib_GraphicsDrawLine_AZ_32(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, mllib_s16 <i>x1</i>, mllib_s16 <i>y1</i>, mllib_s16 <i>z1</i>, mllib_s16 <i>x2</i>, mllib_s16 <i>y2</i>, mllib_s16 <i>z2</i>, mllib_s32 <i>c</i>);</pre>																		
DESCRIPTION	Each of these functions draws a line between (x1,y1) and (x2,y2).																		
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>zbuffer</i></td><td>Pointer to the image that holds the Z buffer.</td></tr><tr><td><i>x1</i></td><td>X coordinate of the first point.</td></tr><tr><td><i>y1</i></td><td>Y coordinate of the first point.</td></tr><tr><td><i>z1</i></td><td>Z coordinate of the first point.</td></tr><tr><td><i>x2</i></td><td>X coordinate of the second point.</td></tr><tr><td><i>y2</i></td><td>Y coordinate of the second point.</td></tr><tr><td><i>z2</i></td><td>Z coordinate of the second point.</td></tr><tr><td><i>c</i></td><td>Color used in the drawing.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>zbuffer</i>	Pointer to the image that holds the Z buffer.	<i>x1</i>	X coordinate of the first point.	<i>y1</i>	Y coordinate of the first point.	<i>z1</i>	Z coordinate of the first point.	<i>x2</i>	X coordinate of the second point.	<i>y2</i>	Y coordinate of the second point.	<i>z2</i>	Z coordinate of the second point.	<i>c</i>	Color used in the drawing.
<i>buffer</i>	Pointer to the image into which the function is drawing.																		
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.																		
<i>x1</i>	X coordinate of the first point.																		
<i>y1</i>	Y coordinate of the first point.																		
<i>z1</i>	Z coordinate of the first point.																		
<i>x2</i>	X coordinate of the second point.																		
<i>y2</i>	Y coordinate of the second point.																		
<i>z2</i>	Z coordinate of the second point.																		
<i>c</i>	Color used in the drawing.																		
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.																		
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe												
ATTRIBUTE TYPE	ATTRIBUTE VALUE																		
Interface Stability	Evolving																		
MT-Level	MT-Safe																		
SEE ALSO	mllib_GraphicsDrawLine_8(3MLIB), mllib_GraphicsDrawLine_A_8(3MLIB), mllib_GraphicsDrawLine_AG_8(3MLIB), mllib_GraphicsDrawLine_AGZ_8(3MLIB), mllib_GraphicsDrawLine_G_8(3MLIB), mllib_GraphicsDrawLine_GZ_8(3MLIB), mllib_GraphicsDrawLine_X_8(3MLIB), mllib_GraphicsDrawLine_Z_8(3MLIB), attributes(5)																		

mllib_GraphicsDrawLineFanSet_8(3MLIB)

NAME mllib_GraphicsDrawLineFanSet_8, mllib_GraphicsDrawLineFanSet_32 – draw line set where all members of the set have one common end point

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawLineFanSet_8(mllib_image *buffer, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c);

mllib_status mllib_GraphicsDrawLineFanSet_32(mllib_image *buffer,
    const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
    mllib_s32 c);
```

DESCRIPTION Each of these functions draws a set of lines connecting (x1,y1) with (x2,y2), (x1,y1) with (x3,y3), ..., and (x1,y1) with (xn,yn).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.
x Pointer to array of X coordinates of the points.
y Pointer to array of Y coordinates of the points.
npoints Number of points in the arrays.
c Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawLineFanSet_A_8(3MLIB),
 mllib_GraphicsDrawLineFanSet_AG_8(3MLIB),
 mllib_GraphicsDrawLineFanSet_AGZ_8(3MLIB),
 mllib_GraphicsDrawLineFanSet_AZ_8(3MLIB),
 mllib_GraphicsDrawLineFanSet_G_8(3MLIB),
 mllib_GraphicsDrawLineFanSet_GZ_8(3MLIB),
 mllib_GraphicsDrawLineFanSet_X_8(3MLIB),
 mllib_GraphicsDrawLineFanSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawLineFanSet_A_8(3MLIB)

NAME	mllib_GraphicsDrawLineFanSet_A_8, mllib_GraphicsDrawLineFanSet_A_32 – draw line set with antialiasing, where all members of the set have one common end point										
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawLineFanSet_A_8(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c); mllib_status mllib_GraphicsDrawLineFanSet_A_32(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c);</pre>										
DESCRIPTION	Each of these functions draws a set of lines connecting (x1,y1) with (x2,y2), (x1,y1) with (x3,y3), ..., and (x1,y1) with (xn,yn).										
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>x</i></td><td>Pointer to array of X coordinates of the points.</td></tr><tr><td><i>y</i></td><td>Pointer to array of Y coordinates of the points.</td></tr><tr><td><i>npoints</i></td><td>Number of points in the arrays.</td></tr><tr><td><i>c</i></td><td>Color used in the drawing.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>x</i>	Pointer to array of X coordinates of the points.	<i>y</i>	Pointer to array of Y coordinates of the points.	<i>npoints</i>	Number of points in the arrays.	<i>c</i>	Color used in the drawing.
<i>buffer</i>	Pointer to the image into which the function is drawing.										
<i>x</i>	Pointer to array of X coordinates of the points.										
<i>y</i>	Pointer to array of Y coordinates of the points.										
<i>npoints</i>	Number of points in the arrays.										
<i>c</i>	Color used in the drawing.										
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.										
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe				
ATTRIBUTE TYPE	ATTRIBUTE VALUE										
Interface Stability	Evolving										
MT-Level	MT-Safe										
SEE ALSO	mllib_GraphicsDrawLineFanSet_8(3MLIB), mllib_GraphicsDrawLineFanSet_AG_8(3MLIB), mllib_GraphicsDrawLineFanSet_AGZ_8(3MLIB), mllib_GraphicsDrawLineFanSet_AZ_8(3MLIB), mllib_GraphicsDrawLineFanSet_G_8(3MLIB), mllib_GraphicsDrawLineFanSet_GZ_8(3MLIB), mllib_GraphicsDrawLineFanSet_X_8(3MLIB), mllib_GraphicsDrawLineFanSet_Z_8(3MLIB), attributes(5)										

mllib_GraphicsDrawLineFanSet_AG_8(3MLIB)

NAME mllib_GraphicsDrawLineFanSet_AG_8, mllib_GraphicsDrawLineFanSet_AG_32 – draw line set with antialiasing and gouraud shading, where all members of the set have one common end point

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawLineFanSet_AG_8(mllib_image *buffer,
        const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, const
        mllib_s32 *c);

mllib_status mllib_GraphicsDrawLineFanSet_AG_32(mllib_image *buffer,
        const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, const
        mllib_s32 *c);
```

DESCRIPTION Each of these functions draws a set of lines connecting (x1,y1) with (x2,y2), (x1,y1) with (x3,y3), ..., and (x1,y1) with (xn,yn).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.
x Pointer to array of X coordinates of the points.
y Pointer to array of Y coordinates of the points.
npoints Number of points in the arrays.
c Pointer to array of colors of the points.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawLineFanSet_8(3MLIB),
 mllib_GraphicsDrawLineFanSet_A_8(3MLIB),
 mllib_GraphicsDrawLineFanSet_AGZ_8(3MLIB),
 mllib_GraphicsDrawLineFanSet_AZ_8(3MLIB),
 mllib_GraphicsDrawLineFanSet_G_8(3MLIB),
 mllib_GraphicsDrawLineFanSet_GZ_8(3MLIB),
 mllib_GraphicsDrawLineFanSet_X_8(3MLIB),
 mllib_GraphicsDrawLineFanSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawLineFanSet_AGZ_8(3MLIB)

NAME	mllib_GraphicsDrawLineFanSet_AGZ_8, mllib_GraphicsDrawLineFanSet_AGZ_32 – draw line set with antialiasing, Gouraud shading, and Z buffering, where all members of the set have one common end point						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawLineFanSet_AGZ_8(mllib_image *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c); mllib_status mllib_GraphicsDrawLineFanSet_AGZ_32(mllib_image *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);</pre>						
DESCRIPTION	Each of these functions draws a set of lines connecting (x1,y1) with (x2,y2), (x1,y1) with (x3,y3), ..., and (x1,y1) with (xn,yn).						
PARAMETERS	Each of the functions takes the following arguments: <i>buffer</i> Pointer to the image into which the function is drawing. <i>zbuffer</i> Pointer to the image that holds the Z buffer. <i>x</i> Pointer to array of X coordinates of the points. <i>y</i> Pointer to array of Y coordinates of the points. <i>z</i> Pointer to array of Z coordinates of the points. <i>npoints</i> Number of points in the arrays. <i>c</i> Pointer to array of colors of the points.						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_GraphicsDrawLineFanSet_8(3MLIB), mllib_GraphicsDrawLineFanSet_A_8(3MLIB), mllib_GraphicsDrawLineFanSet_AG_8(3MLIB), mllib_GraphicsDrawLineFanSet_AZ_8(3MLIB), mllib_GraphicsDrawLineFanSet_G_8(3MLIB), mllib_GraphicsDrawLineFanSet_GZ_8(3MLIB), mllib_GraphicsDrawLineFanSet_X_8(3MLIB), mllib_GraphicsDrawLineFanSet_Z_8(3MLIB), attributes(5)						

mllib_GraphicsDrawLineFanSet_AZ_8(3MLIB)

NAME mllib_GraphicsDrawLineFanSet_AZ_8, mllib_GraphicsDrawLineFanSet_AZ_32 – draw line set with antialiasing and Z buffering, where all members of the set have one common end point

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawLineFanSet_AZ_8(mllib_image *buffer,
        mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const
        mllib_s16 *z, mllib_s32 npoints, mllib_s32 c);

mllib_status mllib_GraphicsDrawLineFanSet_AZ_32(mllib_image *buffer,
        mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const
        mllib_s16 *z, mllib_s32 npoints, mllib_s32 c);
```

DESCRIPTION Each of these functions draws a set of lines connecting (x1,y1) with (x2,y2), (x1,y1) with (x3,y3), ..., and (x1,y1) with (xn,yn).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

zbuffer Pointer to the image that holds the Z buffer.

x Pointer to array of X coordinates of the points.

y Pointer to array of Y coordinates of the points.

z Pointer to array of Z coordinates of the points.

npoints Number of points in the arrays.

c Color used in drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawLineFanSet_8(3MLIB), mllib_GraphicsDrawLineFanSet_A_8(3MLIB), mllib_GraphicsDrawLineFanSet_AG_8(3MLIB), mllib_GraphicsDrawLineFanSet_AGZ_8(3MLIB), mllib_GraphicsDrawLineFanSet_G_8(3MLIB), mllib_GraphicsDrawLineFanSet_GZ_8(3MLIB), mllib_GraphicsDrawLineFanSet_X_8(3MLIB), mllib_GraphicsDrawLineFanSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawLineFanSet_G_8(3MLIB)

NAME	mllib_GraphicsDrawLineFanSet_G_8, mllib_GraphicsDrawLineFanSet_G_32 – draw line set with Gouraud shading, where all members of the set have one common end point						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawLineFanSet_G_8(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, const mllib_s32 *<i>c</i>); mllib_status mllib_GraphicsDrawLineFanSet_G_32(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, const mllib_s32 *<i>c</i>);</pre>						
DESCRIPTION	Each of these functions draws a set of lines connecting (x1,y1) with (x2,y2), (x1,y1) with (x3,y3), ..., and (x1,y1) with (xn,yn).						
PARAMETERS	Each of the functions takes the following arguments: <i>buffer</i> Pointer to the image into which the function is drawing. <i>x</i> Pointer to array of X coordinates of the points. <i>y</i> Pointer to array of Y coordinates of the points. <i>npoints</i> Number of points in the arrays. <i>c</i> Pointer to array of colors of the points.						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_GraphicsDrawLineFanSet_8(3MLIB), mllib_GraphicsDrawLineFanSet_A_8(3MLIB), mllib_GraphicsDrawLineFanSet_AG_8(3MLIB), mllib_GraphicsDrawLineFanSet_AGZ_8(3MLIB), mllib_GraphicsDrawLineFanSet_AZ_8(3MLIB), mllib_GraphicsDrawLineFanSet_GZ_8(3MLIB), mllib_GraphicsDrawLineFanSet_X_8(3MLIB), mllib_GraphicsDrawLineFanSet_Z_8(3MLIB), attributes(5)						

mllib_GraphicsDrawLineFanSet_GZ_8(3MLIB)

NAME mllib_GraphicsDrawLineFanSet_GZ_8, mllib_GraphicsDrawLineFanSet_GZ_32 – draw line set with Gouraud shading and Z buffering, where all members of the set have one common end point

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawLineFanSet_GZ_8(mllib_image *buffer,
        mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const
        mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);

mllib_status mllib_GraphicsDrawLineFanSet_GZ_32(mllib_image *buffer,
        mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const
        mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);
```

DESCRIPTION Each of these functions draws a set of lines connecting (x1,y1) with (x2,y2), (x1,y1) with (x3,y3), ..., and (x1,y1) with (xn,yn).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.
zbuffer Pointer to the image that holds the Z buffer.
x Pointer to array of X coordinates of the points.
y Pointer to array of Y coordinates of the points.
z Pointer to array of Z coordinates of the points.
npoints Number of points in the arrays.
c Pointer to array of colors of the points.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawLineFanSet_8(3MLIB),
mllib_GraphicsDrawLineFanSet_A_8(3MLIB),
mllib_GraphicsDrawLineFanSet_AG_8(3MLIB),
mllib_GraphicsDrawLineFanSet_AGZ_8(3MLIB),
mllib_GraphicsDrawLineFanSet_AZ_8(3MLIB),
mllib_GraphicsDrawLineFanSet_G_8(3MLIB),
mllib_GraphicsDrawLineFanSet_X_8(3MLIB),
mllib_GraphicsDrawLineFanSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawLineFanSet_X_8(3MLIB)

NAME	mllib_GraphicsDrawLineFanSet_X_8, mllib_GraphicsDrawLineFanSet_X_32 – draw line set in Xor mode where all members of the set have one common end point												
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawLineFanSet_X_8(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c, mllib_s32 c2); mllib_status mllib_GraphicsDrawLineFanSet_X_32(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c, mllib_s32 c2);</pre>												
DESCRIPTION	Each of these functions draws a set of lines connecting (x1,y1) with (x2,y2), (x1,y1) with (x3,y3), ..., and (x1,y1) with (xn,yn).												
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>x</i></td><td>Pointer to array of X coordinates of the points.</td></tr><tr><td><i>y</i></td><td>Pointer to array of Y coordinates of the points.</td></tr><tr><td><i>npoints</i></td><td>Number of points in the arrays.</td></tr><tr><td><i>c</i></td><td>Color used in the drawing.</td></tr><tr><td><i>c2</i></td><td>Alternation color.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>x</i>	Pointer to array of X coordinates of the points.	<i>y</i>	Pointer to array of Y coordinates of the points.	<i>npoints</i>	Number of points in the arrays.	<i>c</i>	Color used in the drawing.	<i>c2</i>	Alternation color.
<i>buffer</i>	Pointer to the image into which the function is drawing.												
<i>x</i>	Pointer to array of X coordinates of the points.												
<i>y</i>	Pointer to array of Y coordinates of the points.												
<i>npoints</i>	Number of points in the arrays.												
<i>c</i>	Color used in the drawing.												
<i>c2</i>	Alternation color.												
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.												
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe						
ATTRIBUTE TYPE	ATTRIBUTE VALUE												
Interface Stability	Evolving												
MT-Level	MT-Safe												
SEE ALSO	mllib_GraphicsDrawLineFanSet_8(3MLIB), mllib_GraphicsDrawLineFanSet_A_8(3MLIB), mllib_GraphicsDrawLineFanSet_AG_8(3MLIB), mllib_GraphicsDrawLineFanSet_AGZ_8(3MLIB), mllib_GraphicsDrawLineFanSet_AZ_8(3MLIB), mllib_GraphicsDrawLineFanSet_G_8(3MLIB), mllib_GraphicsDrawLineFanSet_GZ_8(3MLIB), mllib_GraphicsDrawLineFanSet_Z_8(3MLIB), attributes(5)												

mllib_GraphicsDrawLineFanSet_Z_8(3MLIB)

NAME | mllib_GraphicsDrawLineFanSet_Z_8, mllib_GraphicsDrawLineFanSet_Z_32 – draw line set with Z buffering where all members of the set have one common end point

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawLineFanSet_Z_8(mllib_image *buffer,
        mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const
        mllib_s16 *z, mllib_s32 npoints, mllib_s32 c);

mllib_status mllib_GraphicsDrawLineFanSet_Z_32(mllib_image *buffer,
        mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const
        mllib_s16 *z, mllib_s32 npoints, mllib_s32 c);
```

DESCRIPTION | Each of these functions draws a set of lines connecting (x1,y1) with (x2,y2), (x1,y1) with (x3,y3), ..., and (x1,y1) with (xn,yn).

PARAMETERS | Each of the functions takes the following arguments:

buffer | Pointer to the image into which the function is drawing.

zbuffer | Pointer to the image that holds the Z buffer.

x | Pointer to array of X coordinates of the points.

y | Pointer to array of Y coordinates of the points.

z | Pointer to array of Z coordinates of the points.

npoints | Number of points in the arrays.

c | Color used in the drawing.

RETURN VALUES | Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_GraphicsDrawLineFanSet_8(3MLIB), mllib_GraphicsDrawLineFanSet_A_8(3MLIB), mllib_GraphicsDrawLineFanSet_AG_8(3MLIB), mllib_GraphicsDrawLineFanSet_AGZ_8(3MLIB), mllib_GraphicsDrawLineFanSet_AZ_8(3MLIB), mllib_GraphicsDrawLineFanSet_G_8(3MLIB), mllib_GraphicsDrawLineFanSet_GZ_8(3MLIB), mllib_GraphicsDrawLineFanSet_X_8(3MLIB), attributes(5)

mllib_GraphicsDrawLine_G_8(3MLIB)

NAME	mllib_GraphicsDrawLine_G_8, mllib_GraphicsDrawLine_G_32 – draw line with Gouraud shading														
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawLine_G_8(mllib_image *<i>buffer</i>, mllib_s16 <i>x1</i>, mllib_s16 <i>y1</i>, mllib_s16 <i>x2</i>, mllib_s16 <i>y2</i>, mllib_s32 <i>c1</i>, mllib_s32 <i>c2</i>); mllib_status mllib_GraphicsDrawLine_G_32(mllib_image *<i>buffer</i>, mllib_s16 <i>x1</i>, mllib_s16 <i>y1</i>, mllib_s16 <i>x2</i>, mllib_s16 <i>y2</i>, mllib_s32 <i>c1</i>, mllib_s32 <i>c2</i>);</pre>														
DESCRIPTION	Each of these functions draws a line between (x1,y1) and (x2,y2).														
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>x1</i></td><td>X coordinate of the first point.</td></tr><tr><td><i>y1</i></td><td>Y coordinate of the first point.</td></tr><tr><td><i>x2</i></td><td>X coordinate of the second point.</td></tr><tr><td><i>y2</i></td><td>Y coordinate of the second point.</td></tr><tr><td><i>c1</i></td><td>Color of the first point.</td></tr><tr><td><i>c2</i></td><td>Color of the second point.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>x1</i>	X coordinate of the first point.	<i>y1</i>	Y coordinate of the first point.	<i>x2</i>	X coordinate of the second point.	<i>y2</i>	Y coordinate of the second point.	<i>c1</i>	Color of the first point.	<i>c2</i>	Color of the second point.
<i>buffer</i>	Pointer to the image into which the function is drawing.														
<i>x1</i>	X coordinate of the first point.														
<i>y1</i>	Y coordinate of the first point.														
<i>x2</i>	X coordinate of the second point.														
<i>y2</i>	Y coordinate of the second point.														
<i>c1</i>	Color of the first point.														
<i>c2</i>	Color of the second point.														
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.														
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe								
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														
MT-Level	MT-Safe														
SEE ALSO	mllib_GraphicsDrawLine_8(3MLIB), mllib_GraphicsDrawLine_A_8(3MLIB), mllib_GraphicsDrawLine_AG_8(3MLIB), mllib_GraphicsDrawLine_AGZ_8(3MLIB), mllib_GraphicsDrawLine_AZ_8(3MLIB), mllib_GraphicsDrawLine_GZ_8(3MLIB), mllib_GraphicsDrawLine_X_8(3MLIB), mllib_GraphicsDrawLine_Z_8(3MLIB), attributes(5)														

NAME mllib_GraphicsDrawLine_GZ_8, mllib_GraphicsDrawLine_GZ_32 – draw line with Gouraud shading and Z buffering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawLine_GZ_8(mllib_image *buffer,
      mllib_image *zbuffer, mllib_s16 x1, mllib_s16 y1, mllib_s16 z1,
      mllib_s16 x2, mllib_s16 y2, mllib_s16 z2, mllib_s32 c1, mllib_s32
      c2);

mllib_status mllib_GraphicsDrawLine_GZ_32(mllib_image *buffer,
      mllib_image *zbuffer, mllib_s16 x1, mllib_s16 y1, mllib_s16 z1,
      mllib_s16 x2, mllib_s16 y2, mllib_s16 z2, mllib_s32 c1, mllib_s32
      c2);
```

DESCRIPTION Each of these functions draws a line between (x1,y1) and (x2,y2).

PARAMETERS Each of the functions takes the following arguments:

<i>buffer</i>	Pointer to the image into which the function is drawing.
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.
<i>x1</i>	X coordinate of the first point.
<i>y1</i>	Y coordinate of the first point.
<i>z1</i>	Z coordinate of the first point.
<i>x2</i>	X coordinate of the second point.
<i>y2</i>	Y coordinate of the second point.
<i>z2</i>	Z coordinate of the second point.
<i>c1</i>	Color of the first point.
<i>c2</i>	Color of the second point.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

mllib_GraphicsDrawLine_GZ_8(3MLIB)

SEE ALSO | mllib_GraphicsDrawLine_8(3MLIB), mllib_GraphicsDrawLine_A_8(3MLIB),
mllib_GraphicsDrawLine_AG_8(3MLIB),
mllib_GraphicsDrawLine_AGZ_8(3MLIB),
mllib_GraphicsDrawLine_AZ_8(3MLIB),
mllib_GraphicsDrawLine_G_8(3MLIB), mllib_GraphicsDrawLine_X_8(3MLIB),
mllib_GraphicsDrawLine_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawLineSet_8(3MLIB)

NAME mllib_GraphicsDrawLineSet_8, mllib_GraphicsDrawLineSet_32 – draw line set where each member can have different end points

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawLineSet_8(mllib_image *buffer, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c);

mllib_status mllib_GraphicsDrawLineSet_32(mllib_image *buffer, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c);
```

DESCRIPTION Each of these functions draws a set of lines connecting (x1,y1) with (x2,y2), (x3,y3) with (x4,y4), ..., and (xn-1,yn-1) with (xn,yn).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.
x Pointer to array of X coordinates of the points.
y Pointer to array of Y coordinates of the points.
npoints Number of points in the arrays. *npoints* must be a multiple of 2.
c Color used in the drawing.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_GraphicsDrawLineSet_A_8(3MLIB)`,
`mllib_GraphicsDrawLineSet_AG_8(3MLIB)`,
`mllib_GraphicsDrawLineSet_AGZ_8(3MLIB)`,
`mllib_GraphicsDrawLineSet_AZ_8(3MLIB)`,
`mllib_GraphicsDrawLineSet_G_8(3MLIB)`,
`mllib_GraphicsDrawLineSet_GZ_8(3MLIB)`,
`mllib_GraphicsDrawLineSet_X_8(3MLIB)`,
`mllib_GraphicsDrawLineSet_Z_8(3MLIB)`, `attributes(5)`

mllib_GraphicsDrawLineSet_A_8(3MLIB)

NAME	mllib_GraphicsDrawLineSet_A_8, mllib_GraphicsDrawLineSet_A_32 – draw line set with antialiasing where each member can have different end points						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawLineSet_A_8(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c); mllib_status mllib_GraphicsDrawLineSet_A_32(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c);</pre>						
DESCRIPTION	Each of these functions draws a set of lines connecting (x1,y1) with (x2,y2), (x3,y3) with (x4,y4), ..., and (xn-1,yn-1) with (xn,yn).						
PARAMETERS	Each of the functions takes the following arguments: <i>buffer</i> Pointer to the image into which the function is drawing. <i>x</i> Pointer to array of X coordinates of the points. <i>y</i> Pointer to array of Y coordinates of the points. <i>npoints</i> Number of points in the arrays. npoints must be a multiple of 2. <i>c</i> Color used in the drawing.						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_GraphicsDrawLineSet_8(3MLIB), mllib_GraphicsDrawLineSet_AG_8(3MLIB), mllib_GraphicsDrawLineSet_AGZ_8(3MLIB), mllib_GraphicsDrawLineSet_AZ_8(3MLIB), mllib_GraphicsDrawLineSet_G_8(3MLIB), mllib_GraphicsDrawLineSet_GZ_8(3MLIB), mllib_GraphicsDrawLineSet_X_8(3MLIB), mllib_GraphicsDrawLineSet_Z_8(3MLIB), attributes(5)						

mllib_GraphicsDrawLineSet_AG_8(3MLIB)

NAME | mllib_GraphicsDrawLineSet_AG_8, mllib_GraphicsDrawLineSet_AG_32 – draw line set with antialiasing and Gouraud shading where each member can have different end points

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawLineSet_AG_8(mllib_image *buffer, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, const
    mllib_s32 *c);

mllib_status mllib_GraphicsDrawLineSet_AG_32(mllib_image *buffer,
    const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, const
    mllib_s32 *c);
```

DESCRIPTION | Each of these functions draws a set of lines connecting (x1,y1) with (x2,y2), (x3,y3) with (x4,y4), ..., and (xn-1,yn-1) with (xn,yn).

PARAMETERS | Each of the functions takes the following arguments:

buffer | Pointer to the image into which the function is drawing.

x | Pointer to array of X coordinates of the points.

y | Pointer to array of Y coordinates of the points.

npoints | Number of points in the arrays. *npoints* must be a multiple of 2.

c | Pointer to array of colors of the points.

RETURN VALUES | Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_GraphicsDrawLineSet_8(3MLIB),
 mllib_GraphicsDrawLineSet_A_8(3MLIB),
 mllib_GraphicsDrawLineSet_AGZ_8(3MLIB),
 mllib_GraphicsDrawLineSet_AZ_8(3MLIB),
 mllib_GraphicsDrawLineSet_G_8(3MLIB),
 mllib_GraphicsDrawLineSet_GZ_8(3MLIB),
 mllib_GraphicsDrawLineSet_X_8(3MLIB),
 mllib_GraphicsDrawLineSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawLineSet_AGZ_8(3MLIB)

NAME	mllib_GraphicsDrawLineSet_AGZ_8, mllib_GraphicsDrawLineSet_AGZ_32 – draw line set with antialiasing, Gouraud shading, and Z buffering, where each member can have different end points														
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawLineSet_AGZ_8(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, const mllib_s16 *<i>z</i>, mllib_s32 <i>npoints</i>, const mllib_s32 *<i>c</i>); mllib_status mllib_GraphicsDrawLineSet_AGZ_32(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, const mllib_s16 *<i>z</i>, mllib_s32 <i>npoints</i>, const mllib_s32 *<i>c</i>);</pre>														
DESCRIPTION	Each of these functions draws a set of lines connecting (x1,y1) with (x2,y2), (x3,y3) with (x4,y4), ..., and (xn-1,yn-1) with (xn,yn).														
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>zbuffer</i></td><td>Pointer to the image that holds the Z buffer.</td></tr><tr><td><i>x</i></td><td>Pointer to array of X coordinates of the points.</td></tr><tr><td><i>y</i></td><td>Pointer to array of Y coordinates of the points.</td></tr><tr><td><i>z</i></td><td>Pointer to array of Z coordinates of the points.</td></tr><tr><td><i>npoints</i></td><td>Number of points in the arrays. <i>npoints</i> must be a multiple of 2.</td></tr><tr><td><i>c</i></td><td>Pointer to array of colors of the points.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>zbuffer</i>	Pointer to the image that holds the Z buffer.	<i>x</i>	Pointer to array of X coordinates of the points.	<i>y</i>	Pointer to array of Y coordinates of the points.	<i>z</i>	Pointer to array of Z coordinates of the points.	<i>npoints</i>	Number of points in the arrays. <i>npoints</i> must be a multiple of 2.	<i>c</i>	Pointer to array of colors of the points.
<i>buffer</i>	Pointer to the image into which the function is drawing.														
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.														
<i>x</i>	Pointer to array of X coordinates of the points.														
<i>y</i>	Pointer to array of Y coordinates of the points.														
<i>z</i>	Pointer to array of Z coordinates of the points.														
<i>npoints</i>	Number of points in the arrays. <i>npoints</i> must be a multiple of 2.														
<i>c</i>	Pointer to array of colors of the points.														
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.														
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe								
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														
MT-Level	MT-Safe														
SEE ALSO	<code>mllib_GraphicsDrawLineSet_8(3MLIB)</code> , <code>mllib_GraphicsDrawLineSet_A_8(3MLIB)</code> , <code>mllib_GraphicsDrawLineSet_AG_8(3MLIB)</code> , <code>mllib_GraphicsDrawLineSet_AZ_8(3MLIB)</code> , <code>mllib_GraphicsDrawLineSet_G_8(3MLIB)</code> , <code>mllib_GraphicsDrawLineSet_GZ_8(3MLIB)</code> , <code>mllib_GraphicsDrawLineSet_X_8(3MLIB)</code> , <code>mllib_GraphicsDrawLineSet_Z_8(3MLIB)</code> , <code>attributes(5)</code>														

mllib_GraphicsDrawLineSet_AZ_8(3MLIB)

NAME mllib_GraphicsDrawLineSet_AZ_8, mllib_GraphicsDrawLineSet_AZ_32 – draw line set with antialiasing and Z buffering, where each member can have different end points

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawLineSet_AZ_8(mllib_image *buffer,
      mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const
      mllib_s16 *z, mllib_s32 npoints, mllib_s32 c);

mllib_status mllib_GraphicsDrawLineSet_AZ_32(mllib_image *buffer,
      mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const
      mllib_s16 *z, mllib_s32 npoints, mllib_s32 c);
```

DESCRIPTION Each of these functions draws set of lines connecting (x1,y1) with (x2,y2), (x3,y3) with (x4,y4), ..., and (xn-1,yn-1) with (xn,yn).

PARAMETERS Each of the functions takes the following arguments:

<i>buffer</i>	Pointer to the image into which the function is drawing.
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.
<i>x</i>	Pointer to array of X coordinates of the points.
<i>y</i>	Pointer to array of Y coordinates of the points.
<i>z</i>	Pointer to array of Z coordinates of the points.
<i>npoints</i>	Number of points in the arrays. <i>npoints</i> must be a multiple of 2.
<i>c</i>	Color used in the drawing.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_GraphicsDrawLineSet_8\(3MLIB\)](#), [mllib_GraphicsDrawLineSet_A_8\(3MLIB\)](#), [mllib_GraphicsDrawLineSet_AG_8\(3MLIB\)](#), [mllib_GraphicsDrawLineSet_AGZ_8\(3MLIB\)](#), [mllib_GraphicsDrawLineSet_G_8\(3MLIB\)](#), [mllib_GraphicsDrawLineSet_GZ_8\(3MLIB\)](#), [mllib_GraphicsDrawLineSet_X_8\(3MLIB\)](#), [mllib_GraphicsDrawLineSet_Z_8\(3MLIB\)](#), [attributes\(5\)](#)

mllib_GraphicsDrawLineSet_G_8(3MLIB)

NAME	mllib_GraphicsDrawLineSet_G_8, mllib_GraphicsDrawLineSet_G_32 – draw line set with Gouraud shading where each member can have different end points						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawLineSet_G_8(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, const mllib_s32 *<i>c</i>); mllib_status mllib_GraphicsDrawLineSet_G_32(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, const mllib_s32 *<i>c</i>);</pre>						
DESCRIPTION	Each of these functions draws set of lines connecting (x1,y1) with (x2,y2), (x3,y3) with (x4,y4), ..., and (xn-1,yn-1) with (xn,yn).						
PARAMETERS	Each of the functions takes the following arguments: <i>buffer</i> Pointer to the image into which the function is drawing. <i>x</i> Pointer to array of X coordinates of the points. <i>y</i> Pointer to array of Y coordinates of the points. <i>npoints</i> Number of points in the arrays. <i>npoints</i> must be a multiple of 2. <i>c</i> Pointer to array of colors of the points.						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_GraphicsDrawLineSet_8(3MLIB), mllib_GraphicsDrawLineSet_A_8(3MLIB), mllib_GraphicsDrawLineSet_AG_8(3MLIB), mllib_GraphicsDrawLineSet_AGZ_8(3MLIB), mllib_GraphicsDrawLineSet_AZ_8(3MLIB), mllib_GraphicsDrawLineSet_GZ_8(3MLIB), mllib_GraphicsDrawLineSet_X_8(3MLIB), mllib_GraphicsDrawLineSet_Z_8(3MLIB), attributes(5)						

mllib_GraphicsDrawLineSet_GZ_8(3MLIB)

NAME mllib_GraphicsDrawLineSet_GZ_8, mllib_GraphicsDrawLineSet_GZ_32 – draw line set with Gouraud shading and Z buffering, where each member can have different end points

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawLineSet_GZ_8(mllib_image *buffer,
      mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const
      mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);

mllib_status mllib_GraphicsDrawLineSet_GZ_32(mllib_image *buffer,
      mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const
      mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);
```

DESCRIPTION Each of these functions draws a set of lines connecting (x1,y1) with (x2,y2), (x3,y3) with (x4,y4), ..., and (xn-1,yn-1) with (xn,yn).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

zbuffer Pointer to the image that holds the Z buffer.

x Pointer to array of X coordinates of the points.

y Pointer to array of Y coordinates of the points.

z Pointer to array of Z coordinates of the points.

npoints Number of points in the arrays. *npoints* must be a multiple of 2.

c Pointer to array of colors of the points.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_GraphicsDrawLineSet_8(3MLIB)`,
`mllib_GraphicsDrawLineSet_A_8(3MLIB)`,
`mllib_GraphicsDrawLineSet_AG_8(3MLIB)`,
`mllib_GraphicsDrawLineSet_AGZ_8(3MLIB)`,
`mllib_GraphicsDrawLineSet_AZ_8(3MLIB)`,
`mllib_GraphicsDrawLineSet_G_8(3MLIB)`,
`mllib_GraphicsDrawLineSet_X_8(3MLIB)`,
`mllib_GraphicsDrawLineSet_Z_8(3MLIB)`, `attributes(5)`

mllib_GraphicsDrawLineSet_X_8(3MLIB)

NAME	mllib_GraphicsDrawLineSet_X_8, mllib_GraphicsDrawLineSet_X_32 – draw line set in Xor mode where each member can have different end points						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawLineSet_X_8(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c, mllib_s32 c2); mllib_status mllib_GraphicsDrawLineSet_X_32(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c, mllib_s32 c2);</pre>						
DESCRIPTION	Each of these functions draws set of lines connecting (x1,y1) with (x2,y2), (x3,y3) with (x4,y4), ..., and (xn-1,yn-1) with (xn,yn).						
PARAMETERS	Each of the functions takes the following arguments: <i>buffer</i> Pointer to the image into which the function is drawing. <i>x</i> Pointer to array of X coordinates of the points. <i>y</i> Pointer to array of Y coordinates of the points. <i>npoints</i> Number of points in the arrays. npoints must be a multiple of 2. <i>c</i> Color used in the drawing. <i>c2</i> Alternation color.						
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_GraphicsDrawLineSet_8(3MLIB), mllib_GraphicsDrawLineSet_A_8(3MLIB), mllib_GraphicsDrawLineSet_AG_8(3MLIB), mllib_GraphicsDrawLineSet_AGZ_8(3MLIB), mllib_GraphicsDrawLineSet_AZ_8(3MLIB), mllib_GraphicsDrawLineSet_G_8(3MLIB), mllib_GraphicsDrawLineSet_GZ_8(3MLIB), mllib_GraphicsDrawLineSet_Z_8(3MLIB), attributes(5)						

mllib_GraphicsDrawLineSet_Z_8(3MLIB)

NAME mllib_GraphicsDrawLineSet_Z_8, mllib_GraphicsDrawLineSet_Z_32 – draw line set with Z buffering where each member can have different end points

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawLineSet_Z_8(mllib_image *buffer,
      mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const
      mllib_s16 *z, mllib_s32 npoints, mllib_s32 c);

mllib_status mllib_GraphicsDrawLineSet_Z_32(mllib_image *buffer,
      mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const
      mllib_s16 *z, mllib_s32 npoints, mllib_s32 c);
```

DESCRIPTION Each of these functions draws set of lines connecting (x1,y1) with (x2,y2), (x3,y3) with (x4,y4), ..., and (xn-1,yn-1) with (xn,yn).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

zbuffer Pointer to the image that holds the Z buffer.

x Pointer to array of X coordinates of the points.

y Pointer to array of Y coordinates of the points.

z Pointer to array of Z coordinates of the points.

npoints Number of points in the arrays. *npoints* must be a multiple of 2.

c Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawLineSet_8(3MLIB), mllib_GraphicsDrawLineSet_A_8(3MLIB), mllib_GraphicsDrawLineSet_AG_8(3MLIB), mllib_GraphicsDrawLineSet_AGZ_8(3MLIB), mllib_GraphicsDrawLineSet_AZ_8(3MLIB), mllib_GraphicsDrawLineSet_G_8(3MLIB), mllib_GraphicsDrawLineSet_GZ_8(3MLIB), mllib_GraphicsDrawLineSet_X_8(3MLIB), attributes(5)

mllib_GraphicsDrawLineStripSet_8(3MLIB)

NAME	mllib_GraphicsDrawLineStripSet_8, mllib_GraphicsDrawLineStripSet_32 – draw line set where each member of the set starts at the point where the previous member ended						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawLineStripSet_8(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>); mllib_status mllib_GraphicsDrawLineStripSet_32(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>);</pre>						
DESCRIPTION	Each of these functions draws a set of lines connecting (x1,y1) with (x2,y2), (x2,y2) with (x3, y3), ..., and (xn-1, yn-1) with (xn,yn).						
PARAMETERS	Each of the functions takes the following arguments: <i>buffer</i> Pointer to the image into which the function is drawing. <i>x</i> Pointer to array of X coordinates of the points. <i>y</i> Pointer to array of Y coordinates of the points. <i>npoints</i> Number of points in the arrays. <i>c</i> Color used in the drawing.						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_GraphicsDrawLineStripSet_A_8(3MLIB), mllib_GraphicsDrawLineStripSet_AG_8(3MLIB), mllib_GraphicsDrawLineStripSet_AGZ_8(3MLIB), mllib_GraphicsDrawLineStripSet_AZ_8(3MLIB), mllib_GraphicsDrawLineStripSet_G_8(3MLIB), mllib_GraphicsDrawLineStripSet_GZ_8(3MLIB), mllib_GraphicsDrawLineStripSet_X_8(3MLIB), mllib_GraphicsDrawLineStripSet_Z_8(3MLIB), attributes(5)						

mllib_GraphicsDrawLineStripSet_A_8(3MLIB)

NAME mllib_GraphicsDrawLineStripSet_A_8, mllib_GraphicsDrawLineStripSet_A_32 – draw line set with antialiasing where each member of the set starts at the point where the previous member ended

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawLineStripSet_A_8(mllib_image *buffer,
        const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
        mllib_s32 c);

mllib_status mllib_GraphicsDrawLineStripSet_A_32(mllib_image *buffer,
        const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
        mllib_s32 c);
```

DESCRIPTION Each of these functions draws a set of lines connecting (x1,y1) with (x2,y2), (x2,y2) with (x3,y3), ..., and (xn-1,yn-1) with (xn,yn).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.
x Pointer to array of X coordinates of the points.
y Pointer to array of Y coordinates of the points.
npoints Number of points in the arrays.
c Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawLineStripSet_8(3MLIB),
mllib_GraphicsDrawLineStripSet_AG_8(3MLIB),
mllib_GraphicsDrawLineStripSet_AGZ_8(3MLIB),
mllib_GraphicsDrawLineStripSet_AZ_8(3MLIB),
mllib_GraphicsDrawLineStripSet_G_8(3MLIB),
mllib_GraphicsDrawLineStripSet_GZ_8(3MLIB),
mllib_GraphicsDrawLineStripSet_X_8(3MLIB),
mllib_GraphicsDrawLineStripSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawLineStripSet_AG_8(3MLIB)

NAME	mllib_GraphicsDrawLineStripSet_AG_8, mllib_GraphicsDrawLineStripSet_AG_32 – draw line set with antialiasing and gouraud shading where each member of the set starts at the point where the previous member ended						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawLineStripSet_AG_8(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, const mllib_s32 *c); mllib_status mllib_GraphicsDrawLineStripSet_AG_32(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, const mllib_s32 *c);</pre>						
DESCRIPTION	Each of these functions draws a set of lines connecting (x1,y1) with (x2,y2), (x2,y2) with (x3,y3), ..., and (xn-1,yn-1) with (xn,yn).						
PARAMETERS	Each of the functions takes the following arguments: <i>buffer</i> Pointer to the image into which the function is drawing. <i>x</i> Pointer to array of X coordinates of the points. <i>y</i> Pointer to array of Y coordinates of the points. <i>npoints</i> Number of points in the arrays. <i>c</i> Pointer to array of colors of the points.						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_GraphicsDrawLineStripSet_8(3MLIB), mllib_GraphicsDrawLineStripSet_A_8(3MLIB), mllib_GraphicsDrawLineStripSet_AGZ_8(3MLIB), mllib_GraphicsDrawLineStripSet_AZ_8(3MLIB), mllib_GraphicsDrawLineStripSet_G_8(3MLIB), mllib_GraphicsDrawLineStripSet_GZ_8(3MLIB), mllib_GraphicsDrawLineStripSet_X_8(3MLIB), mllib_GraphicsDrawLineStripSet_Z_8(3MLIB), attributes(5)						

mllib_GraphicsDrawLineStripSet_AGZ_8(3MLIB)

NAME mllib_GraphicsDrawLineStripSet_AGZ_8, mllib_GraphicsDrawLineStripSet_AGZ_32 – draw line set with antialiasing, Gouraud shading, and Z buffering, where each member of the set starts at the point where the previous member ended

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawLineStripSet_AGZ_8(mllib_image
    *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16
    *y, const mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);

mllib_status mllib_GraphicsDrawLineStripSet_AGZ_32(mllib_image
    *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16
    *y, const mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);
```

DESCRIPTION Each of these functions draws a set of lines connecting (x1,y1) with (x2,y2), (x2,y2) with (x3,y3, ..., and (xn-1,yn-1) with (xn,yn).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

zbuffer Pointer to the image that holds the Z buffer.

x Pointer to array of X coordinates of the points.

y Pointer to array of Y coordinates of the points.

z Pointer to array of Z coordinates of the points.

npoints Number of points in the arrays.

c Pointer to array of colors of the points.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawLineStripSet_8(3MLIB), mllib_GraphicsDrawLineStripSet_A_8(3MLIB), mllib_GraphicsDrawLineStripSet_AG_8(3MLIB), mllib_GraphicsDrawLineStripSet_AZ_8(3MLIB), mllib_GraphicsDrawLineStripSet_G_8(3MLIB), mllib_GraphicsDrawLineStripSet_GZ_8(3MLIB), mllib_GraphicsDrawLineStripSet_X_8(3MLIB), mllib_GraphicsDrawLineStripSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawLineStripSet_AZ_8(3MLIB)

NAME	mllib_GraphicsDrawLineStripSet_AZ_8, mllib_GraphicsDrawLineStripSet_AZ_32 – draw line set with antialiasing and Z buffering, where each member of the set starts at the point where the previous member ended														
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawLineStripSet_AZ_8(mllib_image *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *z, mllib_s32 npoints, mllib_s32 c); mllib_status mllib_GraphicsDrawLineStripSet_AZ_32(mllib_image *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *z, mllib_s32 npoints, mllib_s32 c);</pre>														
DESCRIPTION	Each of these functions draws a set of lines connecting (x1,y1) with (x2,y2), (x2,y2) with (x3,y3, ..., and (xn-1,yn-1) with (xn,yn).														
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>zbuffer</i></td><td>Pointer to the image that holds the Z buffer.</td></tr><tr><td><i>x</i></td><td>Pointer to array of X coordinates of the points.</td></tr><tr><td><i>y</i></td><td>Pointer to array of Y coordinates of the points.</td></tr><tr><td><i>z</i></td><td>Pointer to array of Z coordinates of the points.</td></tr><tr><td><i>npoints</i></td><td>Number of points in the arrays.</td></tr><tr><td><i>c</i></td><td>Color used in the drawing.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>zbuffer</i>	Pointer to the image that holds the Z buffer.	<i>x</i>	Pointer to array of X coordinates of the points.	<i>y</i>	Pointer to array of Y coordinates of the points.	<i>z</i>	Pointer to array of Z coordinates of the points.	<i>npoints</i>	Number of points in the arrays.	<i>c</i>	Color used in the drawing.
<i>buffer</i>	Pointer to the image into which the function is drawing.														
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.														
<i>x</i>	Pointer to array of X coordinates of the points.														
<i>y</i>	Pointer to array of Y coordinates of the points.														
<i>z</i>	Pointer to array of Z coordinates of the points.														
<i>npoints</i>	Number of points in the arrays.														
<i>c</i>	Color used in the drawing.														
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.														
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe								
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														
MT-Level	MT-Safe														
SEE ALSO	mllib_GraphicsDrawLineStripSet_8(3MLIB), mllib_GraphicsDrawLineStripSet_A_8(3MLIB), mllib_GraphicsDrawLineStripSet_AG_8(3MLIB), mllib_GraphicsDrawLineStripSet_AGZ_8(3MLIB), mllib_GraphicsDrawLineStripSet_G_8(3MLIB), mllib_GraphicsDrawLineStripSet_GZ_8(3MLIB), mllib_GraphicsDrawLineStripSet_X_8(3MLIB), mllib_GraphicsDrawLineStripSet_Z_8(3MLIB), attributes(5)														

mllib_GraphicsDrawLineStripSet_G_8(3MLIB)

NAME mllib_GraphicsDrawLineStripSet_G_8, mllib_GraphicsDrawLineStripSet_G_32 – draw line set with Gouraud shading, where each member of the set starts at the point where the previous member ended

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawLineStripSet_G_8(mllib_image *buffer,
        const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, const
        mllib_s32 *c);

mllib_status mllib_GraphicsDrawLineStripSet_G_32(mllib_image *buffer,
        const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, const
        mllib_s32 *c);
```

DESCRIPTION Each of these functions draws a set of lines connecting (x1,y1) with (x2,y2), (x2,y2) with (x3,y3, ..., and (xn-1,yn-1) with (xn,yn).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.
x Pointer to array of X coordinates of the points.
y Pointer to array of Y coordinates of the points.
npoints Number of points in the arrays.
c Pointer to array of colors of the points.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawLineStripSet_8(3MLIB),
mllib_GraphicsDrawLineStripSet_A_8(3MLIB),
mllib_GraphicsDrawLineStripSet_AG_8(3MLIB),
mllib_GraphicsDrawLineStripSet_AGZ_8(3MLIB),
mllib_GraphicsDrawLineStripSet_AZ_8(3MLIB),
mllib_GraphicsDrawLineStripSet_GZ_8(3MLIB),
mllib_GraphicsDrawLineStripSet_X_8(3MLIB),
mllib_GraphicsDrawLineStripSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawLineStripSet_GZ_8(3MLIB)

NAME	mllib_GraphicsDrawLineStripSet_GZ_8, mllib_GraphicsDrawLineStripSet_GZ_32 – draw line set with Gouraud shading and Z buffering, where each member of the set starts at the point where the previous member ended						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawLineStripSet_GZ_8(mllib_image *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c); mllib_status mllib_GraphicsDrawLineStripSet_GZ_32(mllib_image *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);</pre>						
DESCRIPTION	Each of these functions draws a set of lines connecting (x1,y1) with (x2,y2), (x2,y2) with (x3,y3, ..., and (xn-1,yn-1) with (xn,yn).						
PARAMETERS	Each of the functions takes the following arguments: <i>buffer</i> Pointer to the image into which the function is drawing. <i>zbuffer</i> Pointer to the image that holds the Z buffer. <i>x</i> Pointer to array of X coordinates of the points. <i>y</i> Pointer to array of Y coordinates of the points. <i>z</i> Pointer to array of Z coordinates of the points. <i>npoints</i> Number of points in the arrays. <i>c</i> Pointer to array of colors of the points.						
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_GraphicsDrawLineStripSet_8(3MLIB), mllib_GraphicsDrawLineStripSet_A_8(3MLIB), mllib_GraphicsDrawLineStripSet_AG_8(3MLIB), mllib_GraphicsDrawLineStripSet_AGZ_8(3MLIB), mllib_GraphicsDrawLineStripSet_AZ_8(3MLIB), mllib_GraphicsDrawLineStripSet_G_8(3MLIB), mllib_GraphicsDrawLineStripSet_X_8(3MLIB), mllib_GraphicsDrawLineStripSet_Z_8(3MLIB), attributes(5)						

mllib_GraphicsDrawLineStripSet_X_8(3MLIB)

NAME	mllib_GraphicsDrawLineStripSet_X_8, mllib_GraphicsDrawLineStripSet_X_32 – draw line set in Xor mode where each member of the set starts at the point where the previous member ended						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawLineStripSet_X_8(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>, mllib_s32 <i>c2</i>); mllib_status mllib_GraphicsDrawLineStripSet_X_32(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>, mllib_s32 <i>c2</i>);</pre>						
DESCRIPTION	Each of these functions draws a set of lines connecting (x1,y1) with (x2,y2), (x2,y2) with (x3,y3, ..., and (xn-1,yn-1) with (xn,yn).						
PARAMETERS	Each of the functions takes the following arguments: <i>buffer</i> Pointer to the image into which the function is drawing. <i>x</i> Pointer to array of X coordinates of the points. <i>y</i> Pointer to array of Y coordinates of the points. <i>npoints</i> Number of points in the arrays. <i>c</i> Color used in the drawing. <i>c2</i> Alternation color.						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_GraphicsDrawLineStripSet_8(3MLIB), mllib_GraphicsDrawLineStripSet_A_8(3MLIB), mllib_GraphicsDrawLineStripSet_AG_8(3MLIB), mllib_GraphicsDrawLineStripSet_AGZ_8(3MLIB), mllib_GraphicsDrawLineStripSet_AZ_8(3MLIB), mllib_GraphicsDrawLineStripSet_G_8(3MLIB), mllib_GraphicsDrawLineStripSet_GZ_8(3MLIB), mllib_GraphicsDrawLineStripSet_Z_8(3MLIB), attributes(5)						

mllib_GraphicsDrawLineStripSet_Z_8(3MLIB)

NAME	mllib_GraphicsDrawLineStripSet_Z_8, mllib_GraphicsDrawLineStripSet_Z_32 – draw line set with Z buffering where each member of the set starts at the point where the previous member ended														
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawLineStripSet_Z_8(mllib_image *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *z, mllib_s32 npoints, mllib_s32 c); mllib_status mllib_GraphicsDrawLineStripSet_Z_32(mllib_image *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *z, mllib_s32 npoints, mllib_s32 c);</pre>														
DESCRIPTION	Each of these functions draws a set of lines connecting (x1,y1) with (x2,y2), (x2,y2) with (x3,y3, ..., and (xn-1,yn-1) with (xn,yn).														
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>zbuffer</i></td><td>Pointer to the image that holds the Z buffer.</td></tr><tr><td><i>x</i></td><td>Pointer to array of X coordinates of the points.</td></tr><tr><td><i>y</i></td><td>Pointer to array of Y coordinates of the points.</td></tr><tr><td><i>z</i></td><td>Pointer to array of Z coordinates of the points.</td></tr><tr><td><i>npoints</i></td><td>Number of points in the arrays.</td></tr><tr><td><i>c</i></td><td>Color used in the drawing.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>zbuffer</i>	Pointer to the image that holds the Z buffer.	<i>x</i>	Pointer to array of X coordinates of the points.	<i>y</i>	Pointer to array of Y coordinates of the points.	<i>z</i>	Pointer to array of Z coordinates of the points.	<i>npoints</i>	Number of points in the arrays.	<i>c</i>	Color used in the drawing.
<i>buffer</i>	Pointer to the image into which the function is drawing.														
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.														
<i>x</i>	Pointer to array of X coordinates of the points.														
<i>y</i>	Pointer to array of Y coordinates of the points.														
<i>z</i>	Pointer to array of Z coordinates of the points.														
<i>npoints</i>	Number of points in the arrays.														
<i>c</i>	Color used in the drawing.														
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.														
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe								
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														
MT-Level	MT-Safe														
SEE ALSO	mllib_GraphicsDrawLineStripSet_8(3MLIB), mllib_GraphicsDrawLineStripSet_A_8(3MLIB), mllib_GraphicsDrawLineStripSet_AG_8(3MLIB), mllib_GraphicsDrawLineStripSet_AGZ_8(3MLIB), mllib_GraphicsDrawLineStripSet_AZ_8(3MLIB), mllib_GraphicsDrawLineStripSet_G_8(3MLIB), mllib_GraphicsDrawLineStripSet_GZ_8(3MLIB), mllib_GraphicsDrawLineStripSet_X_8(3MLIB), attributes(5)														

mllib_GraphicsDrawLine_X_8(3MLIB)

NAME mllib_GraphicsDrawLine_X_8, mllib_GraphicsDrawLine_X_32 – draw line in Xor mode

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawLine_X_8(mllib_image *buffer, mllib_s16
    x1, mllib_s16 y1, mllib_s16 x2, mllib_s16 y2, mllib_s32 c, mllib_s32
    c2);

mllib_status mllib_GraphicsDrawLine_X_32(mllib_image *buffer, mllib_s16
    x1, mllib_s16 y1, mllib_s16 x2, mllib_s16 y2, mllib_s32 c, mllib_s32
    c2);
```

DESCRIPTION Each of these functions draws a line between (x1,y1) and (x2,y2).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

x1 X coordinate of the first point.

y1 Y coordinate of the first point.

x2 X coordinate of the second point.

y2 Y coordinate of the second point.

c Color used in the drawing.

c2 Alternation color.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawLine_8(3MLIB), mllib_GraphicsDrawLine_A_8(3MLIB), mllib_GraphicsDrawLine_AG_8(3MLIB), mllib_GraphicsDrawLine_AGZ_8(3MLIB), mllib_GraphicsDrawLine_AZ_8(3MLIB), mllib_GraphicsDrawLine_G_8(3MLIB), mllib_GraphicsDrawLine_GZ_8(3MLIB), mllib_GraphicsDrawLine_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawLine_Z_8(3MLIB)

NAME	mllib_GraphicsDrawLine_Z_8, mllib_GraphicsDrawLine_Z_32 – draw line with Z buffering																		
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawLine_Z_8(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, mllib_s16 <i>x1</i>, mllib_s16 <i>y1</i>, mllib_s16 <i>z1</i>, mllib_s16 <i>x2</i>, mllib_s16 <i>y2</i>, mllib_s16 <i>z2</i>, mllib_s32 <i>c</i>); mllib_status mllib_GraphicsDrawLine_Z_32(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, mllib_s16 <i>x1</i>, mllib_s16 <i>y1</i>, mllib_s16 <i>z1</i>, mllib_s16 <i>x2</i>, mllib_s16 <i>y2</i>, mllib_s16 <i>z2</i>, mllib_s32 <i>c</i>);</pre>																		
DESCRIPTION	Each of these functions draws a line between (x1,y1) and (x2,y2).																		
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>zbuffer</i></td><td>Pointer to the image that holds the Z buffer.</td></tr><tr><td><i>x1</i></td><td>X coordinate of the first point.</td></tr><tr><td><i>y1</i></td><td>Y coordinate of the first point.</td></tr><tr><td><i>z1</i></td><td>Z coordinate of the first point.</td></tr><tr><td><i>x2</i></td><td>X coordinate of the second point.</td></tr><tr><td><i>y2</i></td><td>Y coordinate of the second point.</td></tr><tr><td><i>z2</i></td><td>Z coordinate of the second point.</td></tr><tr><td><i>c</i></td><td>Color used in the drawing.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>zbuffer</i>	Pointer to the image that holds the Z buffer.	<i>x1</i>	X coordinate of the first point.	<i>y1</i>	Y coordinate of the first point.	<i>z1</i>	Z coordinate of the first point.	<i>x2</i>	X coordinate of the second point.	<i>y2</i>	Y coordinate of the second point.	<i>z2</i>	Z coordinate of the second point.	<i>c</i>	Color used in the drawing.
<i>buffer</i>	Pointer to the image into which the function is drawing.																		
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.																		
<i>x1</i>	X coordinate of the first point.																		
<i>y1</i>	Y coordinate of the first point.																		
<i>z1</i>	Z coordinate of the first point.																		
<i>x2</i>	X coordinate of the second point.																		
<i>y2</i>	Y coordinate of the second point.																		
<i>z2</i>	Z coordinate of the second point.																		
<i>c</i>	Color used in the drawing.																		
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.																		
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe												
ATTRIBUTE TYPE	ATTRIBUTE VALUE																		
Interface Stability	Evolving																		
MT-Level	MT-Safe																		
SEE ALSO	mllib_GraphicsDrawLine_8(3MLIB), mllib_GraphicsDrawLine_A_8(3MLIB), mllib_GraphicsDrawLine_AG_8(3MLIB), mllib_GraphicsDrawLine_AGZ_8(3MLIB), mllib_GraphicsDrawLine_AZ_8(3MLIB), mllib_GraphicsDrawLine_G_8(3MLIB), mllib_GraphicsDrawLine_GZ_8(3MLIB), mllib_GraphicsDrawLine_X_8(3MLIB), attributes(5)																		

mllib_GraphicsDrawPoint_8(3MLIB)

NAME mllib_GraphicsDrawPoint_8, mllib_GraphicsDrawPoint_32 – draw point

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawPoint_8(mllib_image *buffer, mllib_s16
    x, mllib_s16 y, mllib_s32 c);

mllib_status mllib_GraphicsDrawPoint_32(mllib_image *buffer, mllib_s16
    x, mllib_s16 y, mllib_s32 c);
```

DESCRIPTION Each of these functions draws a point at (x,y) in color c.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

x X coordinate of the point.

y Y coordinate of the point.

c Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_GraphicsDrawPoint_X_8\(3MLIB\)](#), [attributes\(5\)](#)

mllib_GraphicsDrawPointSet_8(3MLIB)

NAME	mllib_GraphicsDrawPointSet_8, mllib_GraphicsDrawPointSet_32 – draw point set where each member can be a different point						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawPointSet_8(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>); mllib_status mllib_GraphicsDrawPointSet_32(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>);</pre>						
DESCRIPTION	Each of these functions draws a point set at (x1,y1), (x2,y2), ..., and (xn,yn).						
PARAMETERS	Each of the functions takes the following arguments: <i>buffer</i> Pointer to the image into which the function is drawing. <i>x</i> Pointer to array of X coordinates of the points. <i>y</i> Pointer to array of Y coordinates of the points. <i>npoints</i> Number of points in the arrays. <i>c</i> Color used in the drawing.						
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1" data-bbox="446 1134 1412 1270"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_GraphicsDrawPointSet_X_8(3MLIB) , attributes(5)						

mllib_GraphicsDrawPointSet_X_8(3MLIB)

NAME mllib_GraphicsDrawPointSet_X_8, mllib_GraphicsDrawPointSet_X_32 – draw point set in Xor mode where each member can be a different point

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawPointSet_X_8(mllib_image *buffer, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c,
    mllib_s32 c2);

mllib_status mllib_GraphicsDrawPointSet_X_32(mllib_image *buffer,
    const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
    mllib_s32 c, mllib_s32 c2);
```

DESCRIPTION Each of these functions draws a point set at (x1,y1), (x2,y2), ..., and (xn,yn).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.
x Pointer to array of X coordinates of the points.
y Pointer to array of Y coordinates of the points.
npoints Number of points in the arrays.
c Color used in the drawing.
c2 Alternation color.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawPointSet_8(3MLIB), attributes(5)

mllib_GraphicsDrawPoint_X_8(3MLIB)

NAME mllib_GraphicsDrawPoint_X_8, mllib_GraphicsDrawPoint_X_32 – draw point in Xor mode

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawPoint_X_8(mllib_image *buffer, mllib_s16
    x, mllib_s16 y, mllib_s32 c, mllib_s32 c2);

mllib_status mllib_GraphicsDrawPoint_X_32(mllib_image *buffer,
    mllib_s16 x, mllib_s16 y, mllib_s32 c, mllib_s32 c2);
```

DESCRIPTION Each of these functions draws a point at (x,y) in color c.

PARAMETERS Each of the functions takes the following arguments:

<i>buffer</i>	Pointer to the image into which the function is drawing.
<i>x</i>	X coordinate of the point.
<i>y</i>	Y coordinate of the point.
<i>c</i>	Color used in the drawing.
<i>c2</i>	Alternation color.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawPoint_8(3MLIB), attributes(5)

mllib_GraphicsDrawPolygon_8(3MLIB)

NAME mllib_GraphicsDrawPolygon_8, mllib_GraphicsDrawPolygon_32 – draw polygon

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawPolygon_8(mllib_image *buffer, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c);
mllib_status mllib_GraphicsDrawPolygon_32(mllib_image *buffer, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c);
```

DESCRIPTION Each of these functions draws a polygon enclosing (x1,y1), (x2,y2), ..., and (xn,yn).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

x Pointer to the array of X coordinates of the vertices.

y Pointer to the array of Y coordinates of the vertices.

npoints Number of vertices in the arrays.

c Color used in the drawing.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_GraphicsDrawPolygon_A_8\(3MLIB\)](#),
[mllib_GraphicsDrawPolygon_AG_8\(3MLIB\)](#),
[mllib_GraphicsDrawPolygon_AGZ_8\(3MLIB\)](#),
[mllib_GraphicsDrawPolygon_AZ_8\(3MLIB\)](#),
[mllib_GraphicsDrawPolygon_G_8\(3MLIB\)](#),
[mllib_GraphicsDrawPolygon_GZ_8\(3MLIB\)](#),
[mllib_GraphicsDrawPolygon_X_8\(3MLIB\)](#),
[mllib_GraphicsDrawPolygon_Z_8\(3MLIB\)](#), [attributes\(5\)](#)

mllib_GraphicsDrawPolygon_A_8(3MLIB)

NAME	mllib_GraphicsDrawPolygon_A_8, mllib_GraphicsDrawPolygon_A_32 – draw polygon with antialiasing						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawPolygon_A_8(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>); mllib_status mllib_GraphicsDrawPolygon_A_32(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>);</pre>						
DESCRIPTION	Each of these functions draws a polygon enclosing (x1,y1), (x2,y2), ..., and (xn,yn).						
PARAMETERS	Each of the functions takes the following arguments: <i>buffer</i> Pointer to the image into which the function is drawing. <i>x</i> Pointer to the array of X coordinates of the vertices. <i>y</i> Pointer to the array of Y coordinates of the vertices. <i>npoints</i> Number of vertices in the arrays. <i>c</i> Color used in the drawing.						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_GraphicsDrawPolygon_8(3MLIB) , mllib_GraphicsDrawPolygon_AG_8(3MLIB) , mllib_GraphicsDrawPolygon_AGZ_8(3MLIB) , mllib_GraphicsDrawPolygon_AZ_8(3MLIB) , mllib_GraphicsDrawPolygon_G_8(3MLIB) , mllib_GraphicsDrawPolygon_GZ_8(3MLIB) , mllib_GraphicsDrawPolygon_X_8(3MLIB) , mllib_GraphicsDrawPolygon_Z_8(3MLIB) , attributes(5)						

mllib_GraphicsDrawPolygon_AG_8(3MLIB)

NAME mllib_GraphicsDrawPolygon_AG_8, mllib_GraphicsDrawPolygon_AG_32 – draw line with antialiasing and Gouraud shading

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawPolygon_AG_8(mllib_image *buffer, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, const
    mllib_s32 *c);

mllib_status mllib_GraphicsDrawPolygon_AG_32(mllib_image *buffer,
    const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, const
    mllib_s32 *c);
```

DESCRIPTION Each of these functions draws a polygon enclosing (x1,y1), (x2,y2), ..., and (xn,yn).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.
x Pointer to the array of X coordinates of the vertices.
y Pointer to the array of Y coordinates of the vertices.
npoints Number of vertices in the arrays.
c Pointer to the array of color of the vertices.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawPolygon_8(3MLIB),
 mllib_GraphicsDrawPolygon_A_8(3MLIB),
 mllib_GraphicsDrawPolygon_AGZ_8(3MLIB),
 mllib_GraphicsDrawPolygon_AZ_8(3MLIB),
 mllib_GraphicsDrawPolygon_G_8(3MLIB),
 mllib_GraphicsDrawPolygon_GZ_8(3MLIB),
 mllib_GraphicsDrawPolygon_X_8(3MLIB),
 mllib_GraphicsDrawPolygon_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawPolygon_AGZ_8(3MLIB)

NAME	mllib_GraphicsDrawPolygon_AGZ_8, mllib_GraphicsDrawPolygon_AGZ_32 – draw polygon with antialiasing, Gouraud shading, and Z buffering														
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawPolygon_AGZ_8(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, const mllib_s16 *<i>z</i>, mllib_s32 <i>npoints</i>, const mllib_s32 *<i>c</i>); mllib_status mllib_GraphicsDrawPolygon_AGZ_32(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, const mllib_s16 *<i>z</i>, mllib_s32 <i>npoints</i>, const mllib_s32 *<i>c</i>);</pre>														
DESCRIPTION	Each of these functions draws a polygon enclosing (x1,y1), (x2,y2), ..., and (xn,yn).														
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>zbuffer</i></td><td>Pointer to the image that holds the Z buffer.</td></tr><tr><td><i>x</i></td><td>Pointer to the array of X coordinates of the vertices.</td></tr><tr><td><i>y</i></td><td>Pointer to the array of Y coordinates of the vertices.</td></tr><tr><td><i>z</i></td><td>Pointer to the array of Z coordinates of the vertices.</td></tr><tr><td><i>npoints</i></td><td>Number of vertices in the arrays.</td></tr><tr><td><i>c</i></td><td>Pointer to the array of colors of the vertices.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>zbuffer</i>	Pointer to the image that holds the Z buffer.	<i>x</i>	Pointer to the array of X coordinates of the vertices.	<i>y</i>	Pointer to the array of Y coordinates of the vertices.	<i>z</i>	Pointer to the array of Z coordinates of the vertices.	<i>npoints</i>	Number of vertices in the arrays.	<i>c</i>	Pointer to the array of colors of the vertices.
<i>buffer</i>	Pointer to the image into which the function is drawing.														
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.														
<i>x</i>	Pointer to the array of X coordinates of the vertices.														
<i>y</i>	Pointer to the array of Y coordinates of the vertices.														
<i>z</i>	Pointer to the array of Z coordinates of the vertices.														
<i>npoints</i>	Number of vertices in the arrays.														
<i>c</i>	Pointer to the array of colors of the vertices.														
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.														
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe								
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														
MT-Level	MT-Safe														
SEE ALSO	mllib_GraphicsDrawPolygon_8(3MLIB), mllib_GraphicsDrawPolygon_A_8(3MLIB), mllib_GraphicsDrawPolygon_AG_8(3MLIB), mllib_GraphicsDrawPolygon_AZ_8(3MLIB), mllib_GraphicsDrawPolygon_G_8(3MLIB), mllib_GraphicsDrawPolygon_GZ_8(3MLIB), mllib_GraphicsDrawPolygon_X_8(3MLIB), mllib_GraphicsDrawPolygon_Z_8(3MLIB), attributes(5)														

mllib_GraphicsDrawPolygon_AZ_8(3MLIB)

NAME mllib_GraphicsDrawPolygon_AZ_8, mllib_GraphicsDrawPolygon_AZ_32 – draw polygon with antialiasing and Z buffering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawPolygon_AZ_8(mllib_image *buffer,
      mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const
      mllib_s16 *z, mllib_s32 npoints, mllib_s32 c);

mllib_status mllib_GraphicsDrawPolygon_AZ_32(mllib_image *buffer,
      mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const
      mllib_s16 *z, mllib_s32 npoints, mllib_s32 c);
```

DESCRIPTION Each of these functions draws a polygon enclosing (x1,y1), (x2,y2), ..., and (xn,yn).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.
zbuffer Pointer to the image that holds the Z buffer.
x Pointer to the array of X coordinates of the vertices.
y Pointer to the array of Y coordinates of the vertices.
z Pointer to the array of Z coordinates of the vertices.
npoints Number of vertices in the arrays.
c Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawPolygon_8(3MLIB),
 mllib_GraphicsDrawPolygon_A_8(3MLIB),
 mllib_GraphicsDrawPolygon_AG_8(3MLIB),
 mllib_GraphicsDrawPolygon_AGZ_8(3MLIB),
 mllib_GraphicsDrawPolygon_G_8(3MLIB),
 mllib_GraphicsDrawPolygon_GZ_8(3MLIB),
 mllib_GraphicsDrawPolygon_X_8(3MLIB),
 mllib_GraphicsDrawPolygon_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawPolygon_G_8(3MLIB)

NAME	mllib_GraphicsDrawPolygon_G_8, mllib_GraphicsDrawPolygon_G_32 – draw polygon with Gouraud shading						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawPolygon_G_8(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, const mllib_s32 *<i>c</i>); mllib_status mllib_GraphicsDrawPolygon_G_32(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, const mllib_s32 *<i>c</i>);</pre>						
DESCRIPTION	Each of these functions draws a polygon enclosing (x1,y1), (x2,y2), ..., and (xn,yn).						
PARAMETERS	Each of the functions takes the following arguments: <i>buffer</i> Pointer to the image into which the function is drawing. <i>x</i> Pointer to the array of X coordinates of the vertices. <i>y</i> Pointer to the array of Y coordinates of the vertices. <i>npoints</i> Number of vertices in the arrays. <i>c</i> Pointer to the array of color of the vertices.						
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_GraphicsDrawPolygon_8(3MLIB), mllib_GraphicsDrawPolygon_A_8(3MLIB), mllib_GraphicsDrawPolygon_AG_8(3MLIB), mllib_GraphicsDrawPolygon_AGZ_8(3MLIB), mllib_GraphicsDrawPolygon_AZ_8(3MLIB), mllib_GraphicsDrawPolygon_GZ_8(3MLIB), mllib_GraphicsDrawPolygon_X_8(3MLIB), mllib_GraphicsDrawPolygon_Z_8(3MLIB), attributes(5)						

mllib_GraphicsDrawPolygon_GZ_8(3MLIB)

NAME mllib_GraphicsDrawPolygon_GZ_8, mllib_GraphicsDrawPolygon_GZ_32 – draw polygon with Gouraud shading and Z buffering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawPolygon_GZ_8(mllib_image *buffer,
      mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const
      mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);

mllib_status mllib_GraphicsDrawPolygon_GZ_32(mllib_image *buffer,
      mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const
      mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);
```

DESCRIPTION Each of these functions draws a polygon enclosing (x1,y1), (x2,y2), ..., and (xn,yn).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.
zbuffer Pointer to the image that holds the Z buffer.
x Pointer to the array of X coordinates of the vertices.
y Pointer to the array of Y coordinates of the vertices.
z Pointer to the array of Z coordinates of the vertices.
npoints Number of vertices in the arrays.
c Pointer to the array of colors of the vertices.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawPolygon_8(3MLIB),
 mllib_GraphicsDrawPolygon_A_8(3MLIB),
 mllib_GraphicsDrawPolygon_AG_8(3MLIB),
 mllib_GraphicsDrawPolygon_AGZ_8(3MLIB),
 mllib_GraphicsDrawPolygon_AZ_8(3MLIB),
 mllib_GraphicsDrawPolygon_G_8(3MLIB),
 mllib_GraphicsDrawPolygon_X_8(3MLIB),
 mllib_GraphicsDrawPolygon_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawPolygon_X_8(3MLIB)

NAME	mllib_GraphicsDrawPolygon_X_8, mllib_GraphicsDrawPolygon_X_32 – draw polygon in Xor mode						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawPolygon_X_8(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>, mllib_s32 <i>c2</i>); mllib_status mllib_GraphicsDrawPolygon_X_32(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>, mllib_s32 <i>c2</i>);</pre>						
DESCRIPTION	Each of these functions draws polygon enclosing (x1,y1), (x2,y2), ..., and (xn,yn).						
PARAMETERS	Each of the functions takes the following arguments: <i>buffer</i> Pointer to the image into which the function is drawing. <i>x</i> Pointer to the array of X coordinates of the vertices. <i>y</i> Pointer to the array of Y coordinates of the vertices. <i>npoints</i> Number of vertices in the arrays. <i>c</i> Color used in the drawing. <i>c2</i> Alternation color.						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_GraphicsDrawPolygon_8(3MLIB)</code> , <code>mllib_GraphicsDrawPolygon_A_8(3MLIB)</code> , <code>mllib_GraphicsDrawPolygon_AG_8(3MLIB)</code> , <code>mllib_GraphicsDrawPolygon_AGZ_8(3MLIB)</code> , <code>mllib_GraphicsDrawPolygon_AZ_8(3MLIB)</code> , <code>mllib_GraphicsDrawPolygon_G_8(3MLIB)</code> , <code>mllib_GraphicsDrawPolygon_GZ_8(3MLIB)</code> , <code>mllib_GraphicsDrawPolygon_Z_8(3MLIB)</code> , <code>attributes(5)</code>						

mllib_GraphicsDrawPolygon_Z_8(3MLIB)

NAME mllib_GraphicsDrawPolygon_Z_8, mllib_GraphicsDrawPolygon_Z_32 – draw polygon with Z buffering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawPolygon_Z_8(mllib_image *buffer,
      mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const
      mllib_s16 *z, mllib_s32 npoints, mllib_s32 c);

mllib_status mllib_GraphicsDrawPolygon_Z_32(mllib_image *buffer,
      mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const
      mllib_s16 *z, mllib_s32 npoints, mllib_s32 c);
```

DESCRIPTION Each of these functions draws a polygon enclosing (x1,y1), (x2,y2), ..., and (xn,yn).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.
zbuffer Pointer to the image that holds the Z buffer.
x Pointer to the array of X coordinates of the vertices.
y Pointer to the array of Y coordinates of the vertices.
z Pointer to the array of Z coordinates of the vertices.
npoints Number of vertices in the arrays.
c Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawPolygon_8(3MLIB),
 mllib_GraphicsDrawPolygon_A_8(3MLIB),
 mllib_GraphicsDrawPolygon_AG_8(3MLIB),
 mllib_GraphicsDrawPolygon_AGZ_8(3MLIB),
 mllib_GraphicsDrawPolygon_AZ_8(3MLIB),
 mllib_GraphicsDrawPolygon_G_8(3MLIB),
 mllib_GraphicsDrawPolygon_GZ_8(3MLIB),
 mllib_GraphicsDrawPolygon_X_8(3MLIB), attributes(5)

mllib_GraphicsDrawPolyline_8(3MLIB)

NAME	mllib_GraphicsDrawPolyline_8, mllib_GraphicsDrawPolyline_32 – draw polyline										
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawPolyline_8(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>); mllib_status mllib_GraphicsDrawPolyline_32(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>);</pre>										
DESCRIPTION	Each of these functions draws a polyline connecting (x1,y1), (x2,y2), ..., and (xn,yn).										
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>x</i></td><td>Pointer to the array of X coordinates of the points.</td></tr><tr><td><i>y</i></td><td>Pointer to the array of Y coordinates of the points.</td></tr><tr><td><i>npoints</i></td><td>Number of points in the arrays.</td></tr><tr><td><i>c</i></td><td>Color used in the drawing.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>x</i>	Pointer to the array of X coordinates of the points.	<i>y</i>	Pointer to the array of Y coordinates of the points.	<i>npoints</i>	Number of points in the arrays.	<i>c</i>	Color used in the drawing.
<i>buffer</i>	Pointer to the image into which the function is drawing.										
<i>x</i>	Pointer to the array of X coordinates of the points.										
<i>y</i>	Pointer to the array of Y coordinates of the points.										
<i>npoints</i>	Number of points in the arrays.										
<i>c</i>	Color used in the drawing.										
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.										
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe				
ATTRIBUTE TYPE	ATTRIBUTE VALUE										
Interface Stability	Evolving										
MT-Level	MT-Safe										
SEE ALSO	mllib_GraphicsDrawPolyline_A_8(3MLIB), mllib_GraphicsDrawPolyline_AG_8(3MLIB), mllib_GraphicsDrawPolyline_AGZ_8(3MLIB), mllib_GraphicsDrawPolyline_AZ_8(3MLIB), mllib_GraphicsDrawPolyline_G_8(3MLIB), mllib_GraphicsDrawPolyline_GZ_8(3MLIB), mllib_GraphicsDrawPolyline_X_8(3MLIB), mllib_GraphicsDrawPolyline_Z_8(3MLIB), attributes(5)										

mllib_GraphicsDrawPolyline_A_8(3MLIB)

NAME mllib_GraphicsDrawPolyline_A_8, mllib_GraphicsDrawPolyline_A_32 – draw polyline with antialiasing

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawPolyline_A_8(mllib_image *buffer, const
      mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c);

mllib_status mllib_GraphicsDrawPolyline_A_32(mllib_image *buffer,
      const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
      mllib_s32 c);
```

DESCRIPTION Each of these functions draws a polyline connecting (x1,y1), (x2,y2), ..., and (xn,yn).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

x Pointer to the array of X coordinates of the points.

y Pointer to the array of Y coordinates of the points.

npoints Number of points in the arrays.

c Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawPolyline_8(3MLIB),
 mllib_GraphicsDrawPolyline_AG_8(3MLIB),
 mllib_GraphicsDrawPolyline_AGZ_8(3MLIB),
 mllib_GraphicsDrawPolyline_AZ_8(3MLIB),
 mllib_GraphicsDrawPolyline_G_8(3MLIB),
 mllib_GraphicsDrawPolyline_GZ_8(3MLIB),
 mllib_GraphicsDrawPolyline_X_8(3MLIB),
 mllib_GraphicsDrawPolyline_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawPolyline_AG_8(3MLIB)

NAME	mllib_GraphicsDrawPolyline_AG_8, mllib_GraphicsDrawPolyline_AG_32 – draw line with antialiasing and Gouraud shading						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawPolyline_AG_8(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, const mllib_s32 *c); mllib_status mllib_GraphicsDrawPolyline_AG_32(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, const mllib_s32 *c);</pre>						
DESCRIPTION	Each of these functions draws a polyline connecting (x1,y1), (x2,y2), ..., and (xn,yn).						
PARAMETERS	Each of the functions takes the following arguments: <i>buffer</i> Pointer to the image into which the function is drawing. <i>x</i> Pointer to the array of X coordinates of the points. <i>y</i> Pointer to the array of Y coordinates of the points. <i>npoints</i> Number of points in the arrays. <i>c</i> Pointer to the array of color of the points.						
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_GraphicsDrawPolyline_8(3MLIB), mllib_GraphicsDrawPolyline_A_8(3MLIB), mllib_GraphicsDrawPolyline_AGZ_8(3MLIB), mllib_GraphicsDrawPolyline_AZ_8(3MLIB), mllib_GraphicsDrawPolyline_G_8(3MLIB), mllib_GraphicsDrawPolyline_GZ_8(3MLIB), mllib_GraphicsDrawPolyline_X_8(3MLIB), mllib_GraphicsDrawPolyline_Z_8(3MLIB), attributes(5)						

mllib_GraphicsDrawPolyline_AGZ_8(3MLIB)

NAME mllib_GraphicsDrawPolyline_AGZ_8, mllib_GraphicsDrawPolyline_AGZ_32 – draw polyline with antialiasing, Gouraud shading, and Z buffering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawPolyline_AGZ_8(mllib_image *buffer,
        mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const
        mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);

mllib_status mllib_GraphicsDrawPolyline_AGZ_32(mllib_image *buffer,
        mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const
        mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);
```

DESCRIPTION Each of these functions draws a polyline connecting (x1,y1), (x2,y2), ..., and (xn,yn).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.
zbuffer Pointer to the image that holds the Z buffer.
x Pointer to the array of X coordinates of the points.
y Pointer to the array of Y coordinates of the points.
z Pointer to the array of Z coordinates of the points.
npoints Number of points in the arrays.
c Pointer to the array of color of the points.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawPolyline_8(3MLIB),
mllib_GraphicsDrawPolyline_A_8(3MLIB),
mllib_GraphicsDrawPolyline_AG_8(3MLIB),
mllib_GraphicsDrawPolyline_AZ_8(3MLIB),
mllib_GraphicsDrawPolyline_G_8(3MLIB),
mllib_GraphicsDrawPolyline_GZ_8(3MLIB),
mllib_GraphicsDrawPolyline_X_8(3MLIB),
mllib_GraphicsDrawPolyline_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawPolyline_AZ_8(3MLIB)

NAME	mllib_GraphicsDrawPolyline_AZ_8, mllib_GraphicsDrawPolyline_AZ_32 – draw polyline with antialiasing and Z buffering														
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawPolyline_AZ_8(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, const mllib_s16 *<i>z</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>); mllib_status mllib_GraphicsDrawPolyline_AZ_32(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, const mllib_s16 *<i>z</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>);</pre>														
DESCRIPTION	Each of these functions draws a polyline connecting (x1,y1), (x2,y2), ..., and (xn,yn).														
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>zbuffer</i></td><td>Pointer to the image that holds the Z buffer.</td></tr><tr><td><i>x</i></td><td>Pointer to the array of X coordinates of the points.</td></tr><tr><td><i>y</i></td><td>Pointer to the array of Y coordinates of the points.</td></tr><tr><td><i>z</i></td><td>Pointer to the array of Z coordinates of the points.</td></tr><tr><td><i>npoints</i></td><td>Number of points in the arrays.</td></tr><tr><td><i>c</i></td><td>Color used in the drawing.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>zbuffer</i>	Pointer to the image that holds the Z buffer.	<i>x</i>	Pointer to the array of X coordinates of the points.	<i>y</i>	Pointer to the array of Y coordinates of the points.	<i>z</i>	Pointer to the array of Z coordinates of the points.	<i>npoints</i>	Number of points in the arrays.	<i>c</i>	Color used in the drawing.
<i>buffer</i>	Pointer to the image into which the function is drawing.														
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.														
<i>x</i>	Pointer to the array of X coordinates of the points.														
<i>y</i>	Pointer to the array of Y coordinates of the points.														
<i>z</i>	Pointer to the array of Z coordinates of the points.														
<i>npoints</i>	Number of points in the arrays.														
<i>c</i>	Color used in the drawing.														
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.														
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe								
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														
MT-Level	MT-Safe														
SEE ALSO	mllib_GraphicsDrawPolyline_8(3MLIB), mllib_GraphicsDrawPolyline_A_8(3MLIB), mllib_GraphicsDrawPolyline_AG_8(3MLIB), mllib_GraphicsDrawPolyline_AGZ_8(3MLIB), mllib_GraphicsDrawPolyline_G_8(3MLIB), mllib_GraphicsDrawPolyline_GZ_8(3MLIB), mllib_GraphicsDrawPolyline_X_8(3MLIB), mllib_GraphicsDrawPolyline_Z_8(3MLIB), attributes(5)														

mllib_GraphicsDrawPolyline_G_8(3MLIB)

NAME mllib_GraphicsDrawPolyline_G_8, mllib_GraphicsDrawPolyline_G_32 – draw polyline with Gouraud shading

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawPolyline_G_8(mllib_image *buffer, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, const
    mllib_s32 *c);

mllib_status mllib_GraphicsDrawPolyline_G_32(mllib_image *buffer,
    const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, const
    mllib_s32 *c);
```

DESCRIPTION Each of these functions draws a polyline connecting (x1,y1), (x2,y2), ..., and (xn,yn).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

x Pointer to the array of X coordinates of the points.

y Pointer to the array of Y coordinates of the points.

npoints Number of points in the arrays.

c Pointer to the array of color of the points.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawPolyline_8(3MLIB),
 mllib_GraphicsDrawPolyline_A_8(3MLIB),
 mllib_GraphicsDrawPolyline_AG_8(3MLIB),
 mllib_GraphicsDrawPolyline_AGZ_8(3MLIB),
 mllib_GraphicsDrawPolyline_AZ_8(3MLIB),
 mllib_GraphicsDrawPolyline_GZ_8(3MLIB),
 mllib_GraphicsDrawPolyline_X_8(3MLIB),
 mllib_GraphicsDrawPolyline_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawPolyline_GZ_8(3MLIB)

NAME	mllib_GraphicsDrawPolyline_GZ_8, mllib_GraphicsDrawPolyline_GZ_32 – draw polyline with Gouraud shading and Z buffering														
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawPolyline_GZ_8(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, const mllib_s16 *<i>z</i>, mllib_s32 <i>npoints</i>, const mllib_s32 *<i>c</i>); mllib_status mllib_GraphicsDrawPolyline_GZ_32(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, const mllib_s16 *<i>z</i>, mllib_s32 <i>npoints</i>, const mllib_s32 *<i>c</i>);</pre>														
DESCRIPTION	Each of these functions draws a polyline connecting (x1,y1), (x2,y2), ..., and (xn,yn).														
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>zbuffer</i></td><td>Pointer to the image that holds the Z buffer.</td></tr><tr><td><i>x</i></td><td>Pointer to the array of X coordinates of the points.</td></tr><tr><td><i>y</i></td><td>Pointer to the array of Y coordinates of the points.</td></tr><tr><td><i>z</i></td><td>Pointer to the array of Z coordinates of the points.</td></tr><tr><td><i>npoints</i></td><td>Number of points in the arrays.</td></tr><tr><td><i>c</i></td><td>Pointer to the array of color of the points.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>zbuffer</i>	Pointer to the image that holds the Z buffer.	<i>x</i>	Pointer to the array of X coordinates of the points.	<i>y</i>	Pointer to the array of Y coordinates of the points.	<i>z</i>	Pointer to the array of Z coordinates of the points.	<i>npoints</i>	Number of points in the arrays.	<i>c</i>	Pointer to the array of color of the points.
<i>buffer</i>	Pointer to the image into which the function is drawing.														
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.														
<i>x</i>	Pointer to the array of X coordinates of the points.														
<i>y</i>	Pointer to the array of Y coordinates of the points.														
<i>z</i>	Pointer to the array of Z coordinates of the points.														
<i>npoints</i>	Number of points in the arrays.														
<i>c</i>	Pointer to the array of color of the points.														
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.														
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe								
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														
MT-Level	MT-Safe														
SEE ALSO	mllib_GraphicsDrawPolyline_8(3MLIB), mllib_GraphicsDrawPolyline_A_8(3MLIB), mllib_GraphicsDrawPolyline_AG_8(3MLIB), mllib_GraphicsDrawPolyline_AGZ_8(3MLIB), mllib_GraphicsDrawPolyline_AZ_8(3MLIB), mllib_GraphicsDrawPolyline_G_8(3MLIB), mllib_GraphicsDrawPolyline_X_8(3MLIB), mllib_GraphicsDrawPolyline_Z_8(3MLIB), attributes(5)														

mllib_GraphicsDrawPolyline_X_8(3MLIB)

NAME mllib_GraphicsDrawPolyline_X_8, mllib_GraphicsDrawPolyline_X_32 – draw polyline in Xor mode

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawPolyline_X_8(mllib_image *buffer, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c,
    mllib_s32 c2);

mllib_status mllib_GraphicsDrawPolyline_X_32(mllib_image *buffer,
    const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
    mllib_s32 c, mllib_s32 c2);
```

DESCRIPTION Each of these functions draws a polyline connecting (x1,y1), (x2,y2), ..., and (xn,yn).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

x Pointer to the array of X coordinates of the points.

y Pointer to the array of Y coordinates of the points.

npoints Number of points in the arrays.

c Color used in the drawing.

c2 Alternation color.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawPolyline_8(3MLIB),
 mllib_GraphicsDrawPolyline_A_8(3MLIB),
 mllib_GraphicsDrawPolyline_AG_8(3MLIB),
 mllib_GraphicsDrawPolyline_AGZ_8(3MLIB),
 mllib_GraphicsDrawPolyline_AZ_8(3MLIB),
 mllib_GraphicsDrawPolyline_G_8(3MLIB),
 mllib_GraphicsDrawPolyline_GZ_8(3MLIB),
 mllib_GraphicsDrawPolyline_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawPolyline_Z_8(3MLIB)

NAME	mllib_GraphicsDrawPolyline_Z_8, mllib_GraphicsDrawPolyline_Z_32 – draw polyline with Z buffering														
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawPolyline_Z_8(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, const mllib_s16 *<i>z</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>); mllib_status mllib_GraphicsDrawPolyline_Z_32(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, const mllib_s16 *<i>z</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>);</pre>														
DESCRIPTION	Each of these functions draws a polyline connecting (x1,y1), (x2,y2), ..., and (xn,yn).														
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>zbuffer</i></td><td>Pointer to the image that holds the Z buffer.</td></tr><tr><td><i>x</i></td><td>Pointer to the array of X coordinates of the points.</td></tr><tr><td><i>y</i></td><td>Pointer to the array of Y coordinates of the points.</td></tr><tr><td><i>z</i></td><td>Pointer to the array of Z coordinates of the points.</td></tr><tr><td><i>npoints</i></td><td>Number of points in the arrays.</td></tr><tr><td><i>c</i></td><td>Color used in the drawing.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>zbuffer</i>	Pointer to the image that holds the Z buffer.	<i>x</i>	Pointer to the array of X coordinates of the points.	<i>y</i>	Pointer to the array of Y coordinates of the points.	<i>z</i>	Pointer to the array of Z coordinates of the points.	<i>npoints</i>	Number of points in the arrays.	<i>c</i>	Color used in the drawing.
<i>buffer</i>	Pointer to the image into which the function is drawing.														
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.														
<i>x</i>	Pointer to the array of X coordinates of the points.														
<i>y</i>	Pointer to the array of Y coordinates of the points.														
<i>z</i>	Pointer to the array of Z coordinates of the points.														
<i>npoints</i>	Number of points in the arrays.														
<i>c</i>	Color used in the drawing.														
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.														
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe								
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														
MT-Level	MT-Safe														
SEE ALSO	mllib_GraphicsDrawPolyline_8(3MLIB), mllib_GraphicsDrawPolyline_A_8(3MLIB), mllib_GraphicsDrawPolyline_AG_8(3MLIB), mllib_GraphicsDrawPolyline_AGZ_8(3MLIB), mllib_GraphicsDrawPolyline_AZ_8(3MLIB), mllib_GraphicsDrawPolyline_G_8(3MLIB), mllib_GraphicsDrawPolyline_GZ_8(3MLIB), mllib_GraphicsDrawPolyline_X_8(3MLIB), attributes(5)														

mllib_GraphicsDrawPolypoint_8(3MLIB)

NAME mllib_GraphicsDrawPolypoint_8, mllib_GraphicsDrawPolypoint_32 – draw polypoint

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawPolypoint_8(mllib_image *buffer, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c);

mllib_status mllib_GraphicsDrawPolypoint_32(mllib_image *buffer, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c);
```

DESCRIPTION Each of these functions draws a polypoint at (x1,y1), (x2,y2), ..., and (xn,yn).

PARAMETERS Each of the functions takes the following arguments:

<i>buffer</i>	Pointer to the image into which the function is drawing.
<i>x</i>	Pointer to the array of the X coordinate of the points.
<i>y</i>	Pointer to the array of the Y coordinate of the points.
<i>npoints</i>	Number of points in the arrays.
<i>c</i>	Color used in the drawing.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_GraphicsDrawPolypoint_X_8\(3MLIB\)](#), [attributes\(5\)](#)

mllib_GraphicsDrawPolypoint_X_8(3MLIB)

NAME	mllib_GraphicsDrawPolypoint_X_8, mllib_GraphicsDrawPolypoint_X_32 – draw polypoint in Xor mode												
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawPolypoint_X_8(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>, mllib_s32 <i>c2</i>); mllib_status mllib_GraphicsDrawPolypoint_X_32(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>, mllib_s32 <i>c2</i>);</pre>												
DESCRIPTION	Each of these functions draws a polypoint at (x1,y1), (x2,y2), ..., and (xn,yn).												
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>x</i></td><td>Pointer to the array of the X coordinate of the points.</td></tr><tr><td><i>y</i></td><td>Pointer to the array of the Y coordinate of the points.</td></tr><tr><td><i>npoints</i></td><td>Number of points in the arrays.</td></tr><tr><td><i>c</i></td><td>Color used in the drawing.</td></tr><tr><td><i>c2</i></td><td>Alternation color.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>x</i>	Pointer to the array of the X coordinate of the points.	<i>y</i>	Pointer to the array of the Y coordinate of the points.	<i>npoints</i>	Number of points in the arrays.	<i>c</i>	Color used in the drawing.	<i>c2</i>	Alternation color.
<i>buffer</i>	Pointer to the image into which the function is drawing.												
<i>x</i>	Pointer to the array of the X coordinate of the points.												
<i>y</i>	Pointer to the array of the Y coordinate of the points.												
<i>npoints</i>	Number of points in the arrays.												
<i>c</i>	Color used in the drawing.												
<i>c2</i>	Alternation color.												
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.												
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe						
ATTRIBUTE TYPE	ATTRIBUTE VALUE												
Interface Stability	Evolving												
MT-Level	MT-Safe												
SEE ALSO	mllib_GraphicsDrawPolypoint_8(3MLIB), attributes(5)												

mllib_GraphicsDrawRectangle_8(3MLIB)

NAME mllib_GraphicsDrawRectangle_8, mllib_GraphicsDrawRectangle_32 – draw rectangle

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawRectangle_8(mllib_image *buffer,
      mllib_s16 x, mllib_s16 y, mllib_s32 w, mllib_s32 h, mllib_s32 c);

mllib_status mllib_GraphicsDrawRectangle_32(mllib_image *buffer,
      mllib_s16 x, mllib_s16 y, mllib_s32 w, mllib_s32 h, mllib_s32 c);
```

DESCRIPTION Each of these functions draws a rectangle with the upper-left corner at (*x,y*), width *w*, and height *h*.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

x X coordinate of the upper-left corner of the rectangle.

y Y coordinate of the upper-left corner of the rectangle.

w Width of the rectangle.

h Height of the rectangle.

c Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_GraphicsDrawRectangle_X_8\(3MLIB\)](#), [attributes\(5\)](#)

mllib_GraphicsDrawRectangle_X_8(3MLIB)

NAME	mllib_GraphicsDrawRectangle_X_8, mllib_GraphicsDrawRectangle_X_32 – draw rectangle in Xor mode														
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawRectangle_X_8(mllib_image *<i>buffer</i>, mllib_s16 <i>x</i>, mllib_s16 <i>y</i>, mllib_s32 <i>w</i>, mllib_s32 <i>h</i>, mllib_s32 <i>c</i>, mllib_s32 <i>c2</i>); mllib_status mllib_GraphicsDrawRectangle_X_32(mllib_image *<i>buffer</i>, mllib_s16 <i>x</i>, mllib_s16 <i>y</i>, mllib_s32 <i>w</i>, mllib_s32 <i>h</i>, mllib_s32 <i>c</i>, mllib_s32 <i>c2</i>);</pre>														
DESCRIPTION	Each of these functions draws a rectangle with the upper-left corner at (<i>x,y</i>), width <i>w</i> , and height <i>h</i> .														
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>x</i></td><td>X coordinate of the upper-left corner of the rectangle.</td></tr><tr><td><i>y</i></td><td>Y coordinate of the upper-left corner of the rectangle.</td></tr><tr><td><i>w</i></td><td>Width of the rectangle.</td></tr><tr><td><i>h</i></td><td>Height of the rectangle.</td></tr><tr><td><i>c</i></td><td>Color used in the drawing.</td></tr><tr><td><i>c2</i></td><td>Alternation color.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>x</i>	X coordinate of the upper-left corner of the rectangle.	<i>y</i>	Y coordinate of the upper-left corner of the rectangle.	<i>w</i>	Width of the rectangle.	<i>h</i>	Height of the rectangle.	<i>c</i>	Color used in the drawing.	<i>c2</i>	Alternation color.
<i>buffer</i>	Pointer to the image into which the function is drawing.														
<i>x</i>	X coordinate of the upper-left corner of the rectangle.														
<i>y</i>	Y coordinate of the upper-left corner of the rectangle.														
<i>w</i>	Width of the rectangle.														
<i>h</i>	Height of the rectangle.														
<i>c</i>	Color used in the drawing.														
<i>c2</i>	Alternation color.														
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.														
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe								
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														
MT-Level	MT-Safe														
SEE ALSO	mllib_GraphicsDrawRectangle_8(3MLIB) , <code>attributes(5)</code>														

mllib_GraphicsDrawTriangle_8(3MLIB)

NAME mllib_GraphicsDrawTriangle_8, mllib_GraphicsDrawTriangle_32 – draw triangle

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawTriangle_8(mllib_image *buffer,
      mllib_s16 x1, mllib_s16 y1, mllib_s16 x2, mllib_s16 y2, mllib_s16
      x3, mllib_s16 y3, mllib_s32 c);

mllib_status mllib_GraphicsDrawTriangle_32(mllib_image *buffer,
      mllib_s16 x1, mllib_s16 y1, mllib_s16 x2, mllib_s16 y2, mllib_s16
      x3, mllib_s16 y3, mllib_s32 c);
```

DESCRIPTION Each of these functions draws a triangle with the vertices at (x1,y1), (x2,y2) and (x3,y3).

PARAMETERS Each of the functions takes the following arguments:

<i>buffer</i>	Pointer to the image into which the function is drawing.
<i>x1</i>	X coordinate of the first vertex.
<i>y1</i>	Y coordinate of the first vertex.
<i>x2</i>	X coordinate of the second vertex.
<i>y2</i>	Y coordinate of the second vertex.
<i>x3</i>	X coordinate of the third vertex.
<i>y3</i>	Y coordinate of the third vertex.
<i>c</i>	Color used in the drawing.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawTriangle_A_8(3MLIB), mllib_GraphicsDrawTriangle_AG_8(3MLIB), mllib_GraphicsDrawTriangle_AGZ_8(3MLIB), mllib_GraphicsDrawTriangle_AZ_8(3MLIB), mllib_GraphicsDrawTriangle_G_8(3MLIB), mllib_GraphicsDrawTriangle_GZ_8(3MLIB), mllib_GraphicsDrawTriangle_X_8(3MLIB), mllib_GraphicsDrawTriangle_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawTriangle_A_8(3MLIB)

NAME	mllib_GraphicsDrawTriangle_A_8, mllib_GraphicsDrawTriangle_A_32 – draw triangle with antialiasing																
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawTriangle_A_8(mllib_image *<i>buffer</i>, mllib_s16 <i>x1</i>, mllib_s16 <i>y1</i>, mllib_s16 <i>x2</i>, mllib_s16 <i>y2</i>, mllib_s16 <i>x3</i>, mllib_s16 <i>y3</i>, mllib_s32 <i>c</i>); mllib_status mllib_GraphicsDrawTriangle_A_32(mllib_image *<i>buffer</i>, mllib_s16 <i>x1</i>, mllib_s16 <i>y1</i>, mllib_s16 <i>x2</i>, mllib_s16 <i>y2</i>, mllib_s16 <i>x3</i>, mllib_s16 <i>y3</i>, mllib_s32 <i>c</i>);</pre>																
DESCRIPTION	Each of these functions draws a triangle with the vertices at (<i>x1</i> , <i>y1</i>), (<i>x2</i> , <i>y2</i>), and (<i>x3</i> , <i>y3</i>).																
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>x1</i></td><td>X coordinate of the first vertex.</td></tr><tr><td><i>y1</i></td><td>Y coordinate of the first vertex.</td></tr><tr><td><i>x2</i></td><td>X coordinate of the second vertex.</td></tr><tr><td><i>y2</i></td><td>Y coordinate of the second vertex.</td></tr><tr><td><i>x3</i></td><td>X coordinate of the third vertex.</td></tr><tr><td><i>y3</i></td><td>Y coordinate of the third vertex.</td></tr><tr><td><i>c</i></td><td>Color used in the drawing.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>x1</i>	X coordinate of the first vertex.	<i>y1</i>	Y coordinate of the first vertex.	<i>x2</i>	X coordinate of the second vertex.	<i>y2</i>	Y coordinate of the second vertex.	<i>x3</i>	X coordinate of the third vertex.	<i>y3</i>	Y coordinate of the third vertex.	<i>c</i>	Color used in the drawing.
<i>buffer</i>	Pointer to the image into which the function is drawing.																
<i>x1</i>	X coordinate of the first vertex.																
<i>y1</i>	Y coordinate of the first vertex.																
<i>x2</i>	X coordinate of the second vertex.																
<i>y2</i>	Y coordinate of the second vertex.																
<i>x3</i>	X coordinate of the third vertex.																
<i>y3</i>	Y coordinate of the third vertex.																
<i>c</i>	Color used in the drawing.																
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.																
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe										
ATTRIBUTE TYPE	ATTRIBUTE VALUE																
Interface Stability	Evolving																
MT-Level	MT-Safe																
SEE ALSO	mllib_GraphicsDrawTriangle_8(3MLIB), mllib_GraphicsDrawTriangle_AG_8(3MLIB), mllib_GraphicsDrawTriangle_AGZ_8(3MLIB), mllib_GraphicsDrawTriangle_AZ_8(3MLIB), mllib_GraphicsDrawTriangle_G_8(3MLIB), mllib_GraphicsDrawTriangle_GZ_8(3MLIB), mllib_GraphicsDrawTriangle_X_8(3MLIB), mllib_GraphicsDrawTriangle_Z_8(3MLIB), attributes(5)																

mllib_GraphicsDrawTriangle_AG_8(3MLIB)

NAME mllib_GraphicsDrawTriangle_AG_8, mllib_GraphicsDrawTriangle_AG_32 – draw triangle with antialiasing and Gouraud shading

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawTriangle_AG_8(mllib_image *buffer,
        mllib_s16 x1, mllib_s16 y1, mllib_s16 x2, mllib_s16 y2, mllib_s16
        x3, mllib_s16 y3, mllib_s32 c1, mllib_s32 c2, mllib_s32 c3);

mllib_status mllib_GraphicsDrawTriangle_AG_32(mllib_image *buffer,
        mllib_s16 x1, mllib_s16 y1, mllib_s16 x2, mllib_s16 y2, mllib_s16
        x3, mllib_s16 y3, mllib_s32 c1, mllib_s32 c2, mllib_s32 c3);
```

DESCRIPTION Each of these functions draws a triangle with the vertices at (x1,y1), (x2,y2), and (x3,y3).

PARAMETERS Each of the functions takes the following arguments:

<i>buffer</i>	Pointer to the image into which the function is drawing.
<i>x1</i>	X coordinate of the first vertex.
<i>y1</i>	Y coordinate of the first vertex.
<i>x2</i>	X coordinate of the second vertex.
<i>y2</i>	Y coordinate of the second vertex.
<i>x3</i>	X coordinate of the third vertex.
<i>y3</i>	Y coordinate of the third vertex.
<i>c1</i>	Color of the first vertex.
<i>c2</i>	Color of the second vertex.
<i>c3</i>	Color of the third vertex.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawTriangle_8(3MLIB), mllib_GraphicsDrawTriangle_A_8(3MLIB), mllib_GraphicsDrawTriangle_AGZ_8(3MLIB), mllib_GraphicsDrawTriangle_AZ_8(3MLIB), mllib_GraphicsDrawTriangle_G_8(3MLIB),

`mllib_GraphicsDrawTriangle_AG_8(3MLIB)`

```
mllib_GraphicsDrawTriangle_GZ_8(3MLIB),  
mllib_GraphicsDrawTriangle_X_8(3MLIB),  
mllib_GraphicsDrawTriangle_Z_8(3MLIB), attributes(5)
```

mllib_GraphicsDrawTriangle_AGZ_8(3MLIB)

NAME	mllib_GraphicsDrawTriangle_AGZ_8, mllib_GraphicsDrawTriangle_AGZ_32 – draw triangle with antialiasing, Gouraud shading, and Z buffering																												
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawTriangle_AGZ_8(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, mllib_s16 <i>x1</i>, mllib_s16 <i>y1</i>, mllib_s16 <i>z1</i>, mllib_s16 <i>x2</i>, mllib_s16 <i>y2</i>, mllib_s16 <i>z2</i>, mllib_s16 <i>x3</i>, mllib_s16 <i>y3</i>, mllib_s16 <i>z3</i>, mllib_s32 <i>c1</i>, mllib_s32 <i>c2</i>, mllib_s32 <i>c3</i>); mllib_status mllib_GraphicsDrawTriangle_AGZ_32(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, mllib_s16 <i>x1</i>, mllib_s16 <i>y1</i>, mllib_s16 <i>z1</i>, mllib_s16 <i>x2</i>, mllib_s16 <i>y2</i>, mllib_s16 <i>z2</i>, mllib_s16 <i>x3</i>, mllib_s16 <i>y3</i>, mllib_s16 <i>z3</i>, mllib_s32 <i>c1</i>, mllib_s32 <i>c2</i>, mllib_s32 <i>c3</i>);</pre>																												
DESCRIPTION	Each of these functions draws a triangle with the vertices at (x1,y1), (x2,y2), and (x3,y3).																												
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <table border="0"> <tr> <td style="padding-right: 20px;"><i>buffer</i></td> <td>Pointer to the image into which the function is drawing.</td> </tr> <tr> <td><i>zbuffer</i></td> <td>Pointer to the image that holds the Z buffer.</td> </tr> <tr> <td><i>x1</i></td> <td>X coordinate of the first vertex.</td> </tr> <tr> <td><i>y1</i></td> <td>Y coordinate of the first vertex.</td> </tr> <tr> <td><i>z1</i></td> <td>Z coordinate of the first vertex.</td> </tr> <tr> <td><i>x2</i></td> <td>X coordinate of the second vertex.</td> </tr> <tr> <td><i>y2</i></td> <td>Y coordinate of the second vertex.</td> </tr> <tr> <td><i>z2</i></td> <td>Z coordinate of the second vertex.</td> </tr> <tr> <td><i>x3</i></td> <td>X coordinate of the third vertex.</td> </tr> <tr> <td><i>y3</i></td> <td>Y coordinate of the third vertex.</td> </tr> <tr> <td><i>z3</i></td> <td>Z coordinate of the third vertex.</td> </tr> <tr> <td><i>c1</i></td> <td>Color of the first vertex.</td> </tr> <tr> <td><i>c2</i></td> <td>Color of the second vertex.</td> </tr> <tr> <td><i>c3</i></td> <td>Color of the third vertex.</td> </tr> </table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>zbuffer</i>	Pointer to the image that holds the Z buffer.	<i>x1</i>	X coordinate of the first vertex.	<i>y1</i>	Y coordinate of the first vertex.	<i>z1</i>	Z coordinate of the first vertex.	<i>x2</i>	X coordinate of the second vertex.	<i>y2</i>	Y coordinate of the second vertex.	<i>z2</i>	Z coordinate of the second vertex.	<i>x3</i>	X coordinate of the third vertex.	<i>y3</i>	Y coordinate of the third vertex.	<i>z3</i>	Z coordinate of the third vertex.	<i>c1</i>	Color of the first vertex.	<i>c2</i>	Color of the second vertex.	<i>c3</i>	Color of the third vertex.
<i>buffer</i>	Pointer to the image into which the function is drawing.																												
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.																												
<i>x1</i>	X coordinate of the first vertex.																												
<i>y1</i>	Y coordinate of the first vertex.																												
<i>z1</i>	Z coordinate of the first vertex.																												
<i>x2</i>	X coordinate of the second vertex.																												
<i>y2</i>	Y coordinate of the second vertex.																												
<i>z2</i>	Z coordinate of the second vertex.																												
<i>x3</i>	X coordinate of the third vertex.																												
<i>y3</i>	Y coordinate of the third vertex.																												
<i>z3</i>	Z coordinate of the third vertex.																												
<i>c1</i>	Color of the first vertex.																												
<i>c2</i>	Color of the second vertex.																												
<i>c3</i>	Color of the third vertex.																												
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.																												
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:																												

mllib_GraphicsDrawTriangle_AGZ_8(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

mllib_GraphicsDrawTriangle_8(3MLIB),
mllib_GraphicsDrawTriangle_A_8(3MLIB),
mllib_GraphicsDrawTriangle_AG_8(3MLIB),
mllib_GraphicsDrawTriangle_AZ_8(3MLIB),
mllib_GraphicsDrawTriangle_G_8(3MLIB),
mllib_GraphicsDrawTriangle_GZ_8(3MLIB),
mllib_GraphicsDrawTriangle_X_8(3MLIB),
mllib_GraphicsDrawTriangle_Z_8(3MLIB), attributes(5)

NAME mllib_GraphicsDrawTriangle_AZ_8, mllib_GraphicsDrawTriangle_AZ_32 – draw triangle with antialiasing and Z buffering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawTriangle_AZ_8(mllib_image *buffer,
      mllib_image *zbuffer, mllib_s16 x1, mllib_s16 y1, mllib_s16 z1,
      mllib_s16 x2, mllib_s16 y2, mllib_s16 z2, mllib_s16 x3, mllib_s16
      y3, mllib_s16 z3, mllib_s32 c);

mllib_status mllib_GraphicsDrawTriangle_AZ_32(mllib_image *buffer,
      mllib_image *zbuffer, mllib_s16 x1, mllib_s16 y1, mllib_s16 z1,
      mllib_s16 x2, mllib_s16 y2, mllib_s16 z2, mllib_s16 x3, mllib_s16
      y3, mllib_s16 z3, mllib_s32 c);
```

DESCRIPTION Each of these functions draws a triangle with the vertices at (x1,y1), (x2,y2), and (x3,y3).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

zbuffer Pointer to the image that holds the Z buffer.

x1 X coordinate of the first vertex.

y1 Y coordinate of the first vertex.

z1 Z coordinate of the first vertex.

x2 X coordinate of the second vertex.

y2 Y coordinate of the second vertex.

z2 Z coordinate of the second vertex.

x3 X coordinate of the third vertex.

y3 Y coordinate of the third vertex.

z3 Z coordinate of the third vertex.

c Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

mllib_GraphicsDrawTriangle_AZ_8(3MLIB)

SEE ALSO | mllib_GraphicsDrawTriangle_8(3MLIB),
mllib_GraphicsDrawTriangle_A_8(3MLIB),
mllib_GraphicsDrawTriangle_AG_8(3MLIB),
mllib_GraphicsDrawTriangle_AGZ_8(3MLIB),
mllib_GraphicsDrawTriangle_G_8(3MLIB),
mllib_GraphicsDrawTriangle_GZ_8(3MLIB),
mllib_GraphicsDrawTriangle_X_8(3MLIB),
mllib_GraphicsDrawTriangle_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawTriangleFanSet_8(3MLIB)

NAME mllib_GraphicsDrawTriangleFanSet_8, mllib_GraphicsDrawTriangleFanSet_32 – draw triangle set where all members of the set have a common vertex

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawTriangleFanSet_8(mllib_image *buffer,
    const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
    mllib_s32 c);

mllib_status mllib_GraphicsDrawTriangleFanSet_32(mllib_image *buffer,
    const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
    mllib_s32 c);
```

DESCRIPTION Each of these functions draws a set of triangles with vertices at {(x1,y1),(x2,y2),(x3,y3)}, {(x1,y1),(x3,y3),(x4,y4)}, ..., and {(x1,y1),(xn-1,yn-1),(xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.
x Pointer to array of X coordinates of the points.
y Pointer to array of Y coordinates of the points.
npoints Number of points in the arrays.
c Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawTriangleFanSet_A_8(3MLIB),
mllib_GraphicsDrawTriangleFanSet_AG_8(3MLIB),
mllib_GraphicsDrawTriangleFanSet_AGZ_8(3MLIB),
mllib_GraphicsDrawTriangleFanSet_AZ_8(3MLIB),
mllib_GraphicsDrawTriangleFanSet_G_8(3MLIB),
mllib_GraphicsDrawTriangleFanSet_GZ_8(3MLIB),
mllib_GraphicsDrawTriangleFanSet_X_8(3MLIB),
mllib_GraphicsDrawTriangleFanSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawTriangleFanSet_A_8(3MLIB)

NAME	mllib_GraphicsDrawTriangleFanSet_A_8, mllib_GraphicsDrawTriangleFanSet_A_32 – draw triangle set with antialiasing where all members of the set have a common vertex						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawTriangleFanSet_A_8(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c); mllib_status mllib_GraphicsDrawTriangleFanSet_A_32(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c);</pre>						
DESCRIPTION	Each of these functions draws a set of triangles with vertices at {(x1,y1),(x2,y2),(x3,y3)}, {(x1,y1),(x3,y3),(x4,y4)}, ..., and {(x1,y1),(xn-1,yn-1),(xn,yn)}.						
PARAMETERS	Each of the functions takes the following arguments: <i>buffer</i> Pointer to the image into which the function is drawing. <i>x</i> Pointer to array of X coordinates of the points. <i>y</i> Pointer to array of Y coordinates of the points. <i>npoints</i> Number of points in the arrays. <i>c</i> Color used in the drawing.						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_GraphicsDrawTriangleFanSet_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_AG_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_AGZ_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_AZ_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_G_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_GZ_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_X_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_Z_8(3MLIB), attributes(5)						

mllib_GraphicsDrawTriangleFanSet_AG_8(3MLIB)

NAME mllib_GraphicsDrawTriangleFanSet_AG_8, mllib_GraphicsDrawTriangleFanSet_AG_32
 – draw triangle set with antialiasing and gouraud shading, where all members of the set have a common vertex

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawTriangleFanSet_AG_8(mllib_image
    *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
    const mllib_s32 *c);

mllib_status mllib_GraphicsDrawTriangleFanSet_AG_32(mllib_image
    *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
    const mllib_s32 *c);
```

DESCRIPTION Each of these functions draws a set of triangles with vertices at {(x1,y1),(x2,y2), (x3,y3)}, {(x1,y1),(x3,y3),(x4,y4)}, ..., and {(x1,y1),(xn-1,yn-1),(xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.
x Pointer to array of X coordinates of the points.
y Pointer to array of Y coordinates of the points.
npoints Number of points in the arrays.
c Pointer to array of colors of the points.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawTriangleFanSet_8(3MLIB),
 mllib_GraphicsDrawTriangleFanSet_A_8(3MLIB),
 mllib_GraphicsDrawTriangleFanSet_AGZ_8(3MLIB),
 mllib_GraphicsDrawTriangleFanSet_AZ_8(3MLIB),
 mllib_GraphicsDrawTriangleFanSet_G_8(3MLIB),
 mllib_GraphicsDrawTriangleFanSet_GZ_8(3MLIB),
 mllib_GraphicsDrawTriangleFanSet_X_8(3MLIB),
 mllib_GraphicsDrawTriangleFanSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawTriangleFanSet_AGZ_8(3MLIB)

NAME	mllib_GraphicsDrawTriangleFanSet_AGZ_8, mllib_GraphicsDrawTriangleFanSet_AGZ_32 – draw triangle set with antialiasing, Gouraud shading, and Z buffering, where all members of the set have a common vertex						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawTriangleFanSet_AGZ_8(mllib_image *buffer, mllib_image *zbuffer, const mlib_s16 *x, const mlib_s16 *y, const mlib_s16 *z, mlib_s32 npoints, const mlib_s32 *c); mllib_status mllib_GraphicsDrawTriangleFanSet_AGZ_32(mllib_image *buffer, mllib_image *zbuffer, const mlib_s16 *x, const mlib_s16 *y, const mlib_s16 *z, mlib_s32 npoints, const mlib_s32 *c);</pre>						
DESCRIPTION	Each of these functions draws a set of triangles with vertices at {(x1,y1),(x2,y2),(x3,y3)}, {(x1,y1),(x3,y3),(x4,y4)}, ..., and {(x1,y1),(xn-1,yn-1),(xn,yn)}.						
PARAMETERS	Each of the functions takes the following arguments: <i>buffer</i> Pointer to the image into which the function is drawing. <i>zbuffer</i> Pointer to the image that holds the Z buffer. <i>x</i> Pointer to array of X coordinates of the points. <i>y</i> Pointer to array of Y coordinates of the points. <i>z</i> Pointer to array of Z coordinates of the points. <i>npoints</i> Number of points in the arrays. <i>c</i> Pointer to array of colors of the points.						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_GraphicsDrawTriangleFanSet_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_A_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_AG_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_AZ_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_G_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_GZ_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_X_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_Z_8(3MLIB), attributes(5)						

mllib_GraphicsDrawTriangleFanSet_AZ_8(3MLIB)

NAME mllib_GraphicsDrawTriangleFanSet_AZ_8, mllib_GraphicsDrawTriangleFanSet_AZ_32
 – draw triangle set with antialiasing and Z buffering, where all members of the set have a common vertex

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawTriangleFanSet_AZ_8(mllib_image
    *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16
    *y, const mllib_s16 *z, mllib_s32 npoints, mllib_s32 c);

mllib_status mllib_GraphicsDrawTriangleFanSet_AZ_32(mllib_image
    *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16
    *y, const mllib_s16 *z, mllib_s32 npoints, mllib_s32 c);
```

DESCRIPTION Each of these functions draws a set of triangles with vertices at {(x1,y1),(x2,y2),(x3,y3)}, {(x1,y1),(x3,y3),(x4,y4)}, ..., and {(x1,y1),(xn-1,yn-1),(xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

<i>buffer</i>	Pointer to the image into which the function is drawing.
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.
<i>x</i>	Pointer to array of X coordinates of the points.
<i>y</i>	Pointer to array of Y coordinates of the points.
<i>z</i>	Pointer to array of Z coordinates of the points.
<i>npoints</i>	Number of points in the arrays.
<i>c</i>	Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawTriangleFanSet_8(3MLIB),
 mllib_GraphicsDrawTriangleFanSet_A_8(3MLIB),
 mllib_GraphicsDrawTriangleFanSet_AG_8(3MLIB),
 mllib_GraphicsDrawTriangleFanSet_AGZ_8(3MLIB),
 mllib_GraphicsDrawTriangleFanSet_G_8(3MLIB),
 mllib_GraphicsDrawTriangleFanSet_GZ_8(3MLIB),
 mllib_GraphicsDrawTriangleFanSet_X_8(3MLIB),
 mllib_GraphicsDrawTriangleFanSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawTriangleFanSet_G_8(3MLIB)

NAME	mllib_GraphicsDrawTriangleFanSet_G_8, mllib_GraphicsDrawTriangleFanSet_G_32 – draw triangle set with Gouraud shading where all members of the set have a common vertex						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawTriangleFanSet_G_8(mllib_image *buffer, const mlib_s16 *x, const mlib_s16 *y, mlib_s32 npoints, const mlib_s32 *c); mllib_status mllib_GraphicsDrawTriangleFanSet_G_32(mllib_image *buffer, const mlib_s16 *x, const mlib_s16 *y, mlib_s32 npoints, const mlib_s32 *c);</pre>						
DESCRIPTION	Each of these functions draws a set of triangles with vertices at {(x1,y1),(x2,y2),(x3,y3)}, {(x1,y1),(x3,y3),(x4,y4)}, ..., and {(x1,y1),(xn-1,yn-1),(xn,yn)}.						
PARAMETERS	Each of the functions takes the following arguments: <i>buffer</i> Pointer to the image into which the function is drawing. <i>x</i> Pointer to array of X coordinates of the points. <i>y</i> Pointer to array of Y coordinates of the points. <i>npoints</i> Number of points in the arrays. <i>c</i> Pointer to array of colors of the points.						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_GraphicsDrawTriangleFanSet_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_A_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_AG_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_AGZ_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_AZ_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_GZ_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_X_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_Z_8(3MLIB), attributes(5)						

mllib_GraphicsDrawTriangleFanSet_GZ_8(3MLIB)

NAME mllib_GraphicsDrawTriangleFanSet_GZ_8, mllib_GraphicsDrawTriangleFanSet_GZ_32
 – draw triangle set with Gouraud shading and Z buffering, where all members of the set have a common vertex

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawTriangleFanSet_GZ_8(mllib_image
    *buffer, mllib_image *zbuffer, const mlib_s16 *x, const mlib_s16
    *y, const mlib_s16 *z, mlib_s32 npoints, const mlib_s32 *c);

mllib_status mllib_GraphicsDrawTriangleFanSet_GZ_32(mllib_image
    *buffer, mllib_image *zbuffer, const mlib_s16 *x, const mlib_s16
    *y, const mlib_s16 *z, mlib_s32 npoints, const mlib_s32 *c);
```

DESCRIPTION Each of these functions draws a set of triangles with vertices at {(x1,y1),(x2,y2),(x3,y3)}, {(x1,y1),(x3,y3),(x4,y4)}, ..., and {(x1,y1),(xn-1,yn-1),(xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

<i>buffer</i>	Pointer to the image into which the function is drawing.
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.
<i>x</i>	Pointer to array of X coordinates of the points.
<i>y</i>	Pointer to array of Y coordinates of the points.
<i>z</i>	Pointer to array of Z coordinates of the points.
<i>npoints</i>	Number of points in the arrays.
<i>c</i>	Pointer to array of colors of the points.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawTriangleFanSet_8(3MLIB),
 mllib_GraphicsDrawTriangleFanSet_A_8(3MLIB),
 mllib_GraphicsDrawTriangleFanSet_AG_8(3MLIB),
 mllib_GraphicsDrawTriangleFanSet_AGZ_8(3MLIB),
 mllib_GraphicsDrawTriangleFanSet_AZ_8(3MLIB),
 mllib_GraphicsDrawTriangleFanSet_G_8(3MLIB),
 mllib_GraphicsDrawTriangleFanSet_X_8(3MLIB),
 mllib_GraphicsDrawTriangleFanSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawTriangleFanSet_X_8(3MLIB)

NAME	mllib_GraphicsDrawTriangleFanSet_X_8, mllib_GraphicsDrawTriangleFanSet_X_32 – draw triangle set in Xor mode where all members of the set have a common vertex						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawTriangleFanSet_X_8(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c, mllib_s32 c2); mllib_status mllib_GraphicsDrawTriangleFanSet_X_32(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c, mllib_s32 c2);</pre>						
DESCRIPTION	Each of these functions draws a set of triangles with vertices at {(x1,y1),(x2,y2),(x3,y3)}, {(x1,y1),(x3,y3),(x4,y4)}, ..., and {(x1,y1),(xn-1,yn-1),(xn,yn)}.						
PARAMETERS	Each of the functions takes the following arguments: <i>buffer</i> Pointer to the image into which the function is drawing. <i>x</i> Pointer to array of X coordinates of the points. <i>y</i> Pointer to array of Y coordinates of the points. <i>npoints</i> Number of points in the arrays. <i>c</i> Color used in the drawing. <i>c2</i> Alternation color.						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_GraphicsDrawTriangleFanSet_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_A_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_AG_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_AGZ_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_AZ_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_G_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_GZ_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_Z_8(3MLIB), attributes(5)						

mllib_GraphicsDrawTriangleFanSet_Z_8(3MLIB)

NAME mllib_GraphicsDrawTriangleFanSet_Z_8, mllib_GraphicsDrawTriangleFanSet_Z_32 – draw triangle set with Z buffering where all members of the set have a common vertex

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawTriangleFanSet_Z_8(mllib_image
    *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16
    *y, const mllib_s16 *z, mllib_s32 npoints, mllib_s32 c);

mllib_status mllib_GraphicsDrawTriangleFanSet_Z_32(mllib_image
    *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16
    *y, const mllib_s16 *z, mllib_s32 npoints, mllib_s32 c);
```

DESCRIPTION Each of these functions draws a set of triangles with vertices at {(x1,y1),(x2,y2),(x3,y3)}, {(x1,y1),(x3,y3),(x4,y4)}, ..., and {(x1,y1),(xn-1,yn-1),(xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

<i>buffer</i>	Pointer to the image into which the function is drawing.
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.
<i>x</i>	Pointer to array of X coordinates of the points.
<i>y</i>	Pointer to array of Y coordinates of the points.
<i>z</i>	Pointer to array of Z coordinates of the points.
<i>npoints</i>	Number of points in the arrays.
<i>c</i>	Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawTriangleFanSet_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_A_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_AG_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_AGZ_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_AZ_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_G_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_GZ_8(3MLIB), mllib_GraphicsDrawTriangleFanSet_X_8(3MLIB), attributes(5)

mllib_GraphicsDrawTriangle_G_8(3MLIB)

NAME	mllib_GraphicsDrawTriangle_G_8, mllib_GraphicsDrawTriangle_G_32 – draw triangle with Gouraud shading																				
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawTriangle_G_8(mllib_image *<i>buffer</i>, mllib_s16 <i>x1</i>, mllib_s16 <i>y1</i>, mllib_s16 <i>x2</i>, mllib_s16 <i>y2</i>, mllib_s16 <i>x3</i>, mllib_s16 <i>y3</i>, mllib_s32 <i>c1</i>, mllib_s32 <i>c2</i>, mllib_s32 <i>c3</i>); mllib_status mllib_GraphicsDrawTriangle_G_32(mllib_image *<i>buffer</i>, mllib_s16 <i>x1</i>, mllib_s16 <i>y1</i>, mllib_s16 <i>x2</i>, mllib_s16 <i>y2</i>, mllib_s16 <i>x3</i>, mllib_s16 <i>y3</i>, mllib_s32 <i>c1</i>, mllib_s32 <i>c2</i>, mllib_s32 <i>c3</i>);</pre>																				
DESCRIPTION	Each of these functions draws a triangle with the vertices at (x1,y1), (x2,y2), and (x3,y3).																				
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>x1</i></td><td>X coordinate of the first vertex.</td></tr><tr><td><i>y1</i></td><td>Y coordinate of the first vertex.</td></tr><tr><td><i>x2</i></td><td>X coordinate of the second vertex.</td></tr><tr><td><i>y2</i></td><td>Y coordinate of the second vertex.</td></tr><tr><td><i>x3</i></td><td>X coordinate of the third vertex.</td></tr><tr><td><i>y3</i></td><td>Y coordinate of the third vertex.</td></tr><tr><td><i>c1</i></td><td>Color of the first vertex.</td></tr><tr><td><i>c2</i></td><td>Color of the second vertex.</td></tr><tr><td><i>c3</i></td><td>Color of the third vertex.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>x1</i>	X coordinate of the first vertex.	<i>y1</i>	Y coordinate of the first vertex.	<i>x2</i>	X coordinate of the second vertex.	<i>y2</i>	Y coordinate of the second vertex.	<i>x3</i>	X coordinate of the third vertex.	<i>y3</i>	Y coordinate of the third vertex.	<i>c1</i>	Color of the first vertex.	<i>c2</i>	Color of the second vertex.	<i>c3</i>	Color of the third vertex.
<i>buffer</i>	Pointer to the image into which the function is drawing.																				
<i>x1</i>	X coordinate of the first vertex.																				
<i>y1</i>	Y coordinate of the first vertex.																				
<i>x2</i>	X coordinate of the second vertex.																				
<i>y2</i>	Y coordinate of the second vertex.																				
<i>x3</i>	X coordinate of the third vertex.																				
<i>y3</i>	Y coordinate of the third vertex.																				
<i>c1</i>	Color of the first vertex.																				
<i>c2</i>	Color of the second vertex.																				
<i>c3</i>	Color of the third vertex.																				
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.																				
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe														
ATTRIBUTE TYPE	ATTRIBUTE VALUE																				
Interface Stability	Evolving																				
MT-Level	MT-Safe																				
SEE ALSO	<code>mllib_GraphicsDrawTriangle_8(3MLIB)</code> , <code>mllib_GraphicsDrawTriangle_A_8(3MLIB)</code> , <code>mllib_GraphicsDrawTriangle_AG_8(3MLIB)</code> , <code>mllib_GraphicsDrawTriangle_AGZ_8(3MLIB)</code> , <code>mllib_GraphicsDrawTriangle_AZ_8(3MLIB)</code> ,																				

`mllib_GraphicsDrawTriangle_G_8(3MLIB)`

```
mllib_GraphicsDrawTriangle_GZ_8(3MLIB),  
mllib_GraphicsDrawTriangle_X_8(3MLIB),  
mllib_GraphicsDrawTriangle_Z_8(3MLIB), attributes(5)
```

mllib_GraphicsDrawTriangle_GZ_8(3MLIB)

NAME	mllib_GraphicsDrawTriangle_GZ_8, mllib_GraphicsDrawTriangle_GZ_32 – draw triangle with Gouraud shading and Z buffering																												
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawTriangle_GZ_8(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, mllib_s16 <i>x1</i>, mllib_s16 <i>y1</i>, mllib_s16 <i>z1</i>, mllib_s16 <i>x2</i>, mllib_s16 <i>y2</i>, mllib_s16 <i>z2</i>, mllib_s16 <i>x3</i>, mllib_s16 <i>y3</i>, mllib_s16 <i>z3</i>, mllib_s32 <i>c1</i>, mllib_s32 <i>c2</i>, mllib_s32 <i>c3</i>); mllib_status mllib_GraphicsDrawTriangle_GZ_32(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, mllib_s16 <i>x1</i>, mllib_s16 <i>y1</i>, mllib_s16 <i>z1</i>, mllib_s16 <i>x2</i>, mllib_s16 <i>y2</i>, mllib_s16 <i>z2</i>, mllib_s16 <i>x3</i>, mllib_s16 <i>y3</i>, mllib_s16 <i>z3</i>, mllib_s32 <i>c1</i>, mllib_s32 <i>c2</i>, mllib_s32 <i>c3</i>);</pre>																												
DESCRIPTION	Each of these functions draws a triangle with vertices at (x1,y1), (x2,y2), and (x3,y3).																												
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>zbuffer</i></td><td>Pointer to the image that holds the Z buffer.</td></tr><tr><td><i>x1</i></td><td>X coordinate of the first vertex.</td></tr><tr><td><i>y1</i></td><td>Y coordinate of the first vertex.</td></tr><tr><td><i>z1</i></td><td>Z coordinate of the first vertex.</td></tr><tr><td><i>x2</i></td><td>X coordinate of the second vertex.</td></tr><tr><td><i>y2</i></td><td>Y coordinate of the second vertex.</td></tr><tr><td><i>z2</i></td><td>Z coordinate of the second vertex.</td></tr><tr><td><i>x3</i></td><td>X coordinate of the third vertex.</td></tr><tr><td><i>y3</i></td><td>Y coordinate of the third vertex.</td></tr><tr><td><i>z3</i></td><td>Z coordinate of the third vertex.</td></tr><tr><td><i>c1</i></td><td>Color of the first vertex.</td></tr><tr><td><i>c2</i></td><td>Color of the second vertex.</td></tr><tr><td><i>c3</i></td><td>Color of the third vertex.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>zbuffer</i>	Pointer to the image that holds the Z buffer.	<i>x1</i>	X coordinate of the first vertex.	<i>y1</i>	Y coordinate of the first vertex.	<i>z1</i>	Z coordinate of the first vertex.	<i>x2</i>	X coordinate of the second vertex.	<i>y2</i>	Y coordinate of the second vertex.	<i>z2</i>	Z coordinate of the second vertex.	<i>x3</i>	X coordinate of the third vertex.	<i>y3</i>	Y coordinate of the third vertex.	<i>z3</i>	Z coordinate of the third vertex.	<i>c1</i>	Color of the first vertex.	<i>c2</i>	Color of the second vertex.	<i>c3</i>	Color of the third vertex.
<i>buffer</i>	Pointer to the image into which the function is drawing.																												
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.																												
<i>x1</i>	X coordinate of the first vertex.																												
<i>y1</i>	Y coordinate of the first vertex.																												
<i>z1</i>	Z coordinate of the first vertex.																												
<i>x2</i>	X coordinate of the second vertex.																												
<i>y2</i>	Y coordinate of the second vertex.																												
<i>z2</i>	Z coordinate of the second vertex.																												
<i>x3</i>	X coordinate of the third vertex.																												
<i>y3</i>	Y coordinate of the third vertex.																												
<i>z3</i>	Z coordinate of the third vertex.																												
<i>c1</i>	Color of the first vertex.																												
<i>c2</i>	Color of the second vertex.																												
<i>c3</i>	Color of the third vertex.																												
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.																												
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:																												

mllib_GraphicsDrawTriangle_GZ_8(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

mllib_GraphicsDrawTriangle_8(3MLIB),
mllib_GraphicsDrawTriangle_A_8(3MLIB),
mllib_GraphicsDrawTriangle_AG_8(3MLIB),
mllib_GraphicsDrawTriangle_AGZ_8(3MLIB),
mllib_GraphicsDrawTriangle_AZ_8(3MLIB),
mllib_GraphicsDrawTriangle_G_8(3MLIB),
mllib_GraphicsDrawTriangle_X_8(3MLIB),
mllib_GraphicsDrawTriangle_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawTriangleSet_8(3MLIB)

NAME	mllib_GraphicsDrawTriangleSet_8, mllib_GraphicsDrawTriangleSet_32 – draw triangle set where each member can have different vertices						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawTriangleSet_8(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>); mllib_status mllib_GraphicsDrawTriangleSet_32(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>);</pre>						
DESCRIPTION	Each of these functions draws a set of triangles with vertices at {(x1,y1), (x2,y2), (x3,y3)}, {(x4,y4),(x5,y5),(x6,y6)}, ..., and {(xn-2,yn-2),(xn-1,yn-1),(xn,yn)}.						
PARAMETERS	Each of the functions takes the following arguments: <p><i>buffer</i> Pointer to the image into which the function is drawing.</p> <p><i>x</i> Pointer to array of X coordinates of the points.</p> <p><i>y</i> Pointer to array of Y coordinates of the points.</p> <p><i>npoints</i> Number of points in the arrays. npoints must be a multiple of 3.</p> <p><i>c</i> Color used in the drawing.</p>						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<p>mllib_GraphicsDrawTriangleSet_A_8(3MLIB), mllib_GraphicsDrawTriangleSet_AG_8(3MLIB), mllib_GraphicsDrawTriangleSet_AGZ_8(3MLIB), mllib_GraphicsDrawTriangleSet_AZ_8(3MLIB), mllib_GraphicsDrawTriangleSet_G_8(3MLIB), mllib_GraphicsDrawTriangleSet_GZ_8(3MLIB), mllib_GraphicsDrawTriangleSet_X_8(3MLIB), mllib_GraphicsDrawTriangleSet_Z_8(3MLIB), attributes(5)</p>						

mllib_GraphicsDrawTriangleSet_A_8(3MLIB)

NAME mllib_GraphicsDrawTriangleSet_A_8, mllib_GraphicsDrawTriangleSet_A_32 – draw triangle set with antialiasing where each member can have different vertices

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawTriangleSet_A_8(mllib_image *buffer,
        const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
        mllib_s32 c);

mllib_status mllib_GraphicsDrawTriangleSet_A_32(mllib_image *buffer,
        const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
        mllib_s32 c);
```

DESCRIPTION Each of these functions draws a set of triangles with vertices at {(x1,y1),(x2,y2),(x3,y3)}, {(x4,y4),(x5,y5),(x6,y6)}, ..., and {(xn-2,yn-2),(xn-1,yn-1),(xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

x Pointer to array of X coordinates of the points.

y Pointer to array of Y coordinates of the points.

npoints Number of points in the arrays. npoints must be a multiple of 3.

c Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawTriangleSet_8(3MLIB), mllib_GraphicsDrawTriangleSet_AG_8(3MLIB), mllib_GraphicsDrawTriangleSet_AGZ_8(3MLIB), mllib_GraphicsDrawTriangleSet_AZ_8(3MLIB), mllib_GraphicsDrawTriangleSet_G_8(3MLIB), mllib_GraphicsDrawTriangleSet_GZ_8(3MLIB), mllib_GraphicsDrawTriangleSet_X_8(3MLIB), mllib_GraphicsDrawTriangleSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawTriangleSet_AG_8(3MLIB)

NAME	mllib_GraphicsDrawTriangleSet_AG_8, mllib_GraphicsDrawTriangleSet_AG_32 – draw triangle set with antialiasing and Gouraud shading, where each member can have different vertices						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_GraphicsDrawTriangleSet_AG_8(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, const mllib_s32 *c); mllib_status mllib_GraphicsDrawTriangleSet_AG_32(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, const mllib_s32 *c);</pre>						
DESCRIPTION	Each of these functions draws a set of triangles with vertices at {(x1,y1),(x2,y2),(x3,y3)}, {(x4,y4),(x5,y5),(x6,y6)}, ..., and {(xn-2,yn-2),(xn-1,yn-1),(xn,yn)}.						
PARAMETERS	Each of the functions takes the following arguments: <i>buffer</i> Pointer to the image into which the function is drawing. <i>x</i> Pointer to array of X coordinates of the points. <i>y</i> Pointer to array of Y coordinates of the points. <i>npoints</i> Number of points in the arrays. npoints must be a multiple of 3. <i>c</i> Pointer to array of colors of the points.						
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_GraphicsDrawTriangleSet_8(3MLIB), mllib_GraphicsDrawTriangleSet_A_8(3MLIB), mllib_GraphicsDrawTriangleSet_AGZ_8(3MLIB), mllib_GraphicsDrawTriangleSet_AZ_8(3MLIB), mllib_GraphicsDrawTriangleSet_G_8(3MLIB), mllib_GraphicsDrawTriangleSet_GZ_8(3MLIB), mllib_GraphicsDrawTriangleSet_X_8(3MLIB), mllib_GraphicsDrawTriangleSet_Z_8(3MLIB), attributes(5)						

mllib_GraphicsDrawTriangleSet_AGZ_8(3MLIB)

NAME mllib_GraphicsDrawTriangleSet_AGZ_8, mllib_GraphicsDrawTriangleSet_AGZ_32 – draw triangle set with antialiasing, Gouraud shading, and Z buffering, where each member can have different vertices

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawTriangleSet_AGZ_8(mllib_image *buffer,
    mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const
    mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);

mllib_status mllib_GraphicsDrawTriangleSet_AGZ_32(mllib_image
    *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16
    *y, const mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);
```

DESCRIPTION Each of these functions draws a set of triangles with vertices at {(x1,y1),(x2,y2),(x3,y3)}, {(x4,y4),(x5,y5),(x6,y6)}, ..., and {(xn-2,yn-2),(xn-1,yn-1),(xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

zbuffer Pointer to the image that holds the Z buffer.

x Pointer to array of X coordinates of the points.

y Pointer to array of Y coordinates of the points.

z Pointer to array of Z coordinates of the points.

npoints Number of points in the arrays. npoints must be a multiple of 3.

c Pointer to array of colors of the points.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawTriangleSet_8(3MLIB), mllib_GraphicsDrawTriangleSet_A_8(3MLIB), mllib_GraphicsDrawTriangleSet_AG_8(3MLIB), mllib_GraphicsDrawTriangleSet_AZ_8(3MLIB), mllib_GraphicsDrawTriangleSet_G_8(3MLIB), mllib_GraphicsDrawTriangleSet_GZ_8(3MLIB), mllib_GraphicsDrawTriangleSet_X_8(3MLIB), mllib_GraphicsDrawTriangleSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawTriangleSet_AZ_8(3MLIB)

NAME	mllib_GraphicsDrawTriangleSet_AZ_8, mllib_GraphicsDrawTriangleSet_AZ_32 – draw triangle set with antialiasing and Z buffering, where each member can have different vertices														
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawTriangleSet_AZ_8(mllib_image *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *z, mllib_s32 npoints, mllib_s32 c); mllib_status mllib_GraphicsDrawTriangleSet_AZ_32(mllib_image *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *z, mllib_s32 npoints, mllib_s32 c);</pre>														
DESCRIPTION	Each of these functions draws set of triangles with vertices at {(x1,y1),(x2,y2), (x3,y3)}, {(x4,y4),(x5,y5),(x6,y6)}, ..., and {(xn-2,yn-2),(xn-1,yn-1),(xn,yn)}.														
PARAMETERS	Each of the functions takes the following arguments: <table border="0" style="margin-left: 20px;"> <tr> <td style="padding-right: 20px;"><i>buffer</i></td> <td>Pointer to the image into which the function is drawing.</td> </tr> <tr> <td><i>zbuffer</i></td> <td>Pointer to the image that holds the Z buffer.</td> </tr> <tr> <td><i>x</i></td> <td>Pointer to array of X coordinates of the points.</td> </tr> <tr> <td><i>y</i></td> <td>Pointer to array of Y coordinates of the points.</td> </tr> <tr> <td><i>z</i></td> <td>Pointer to array of Z coordinates of the points.</td> </tr> <tr> <td><i>npoints</i></td> <td>Number of points in the arrays. npoints must be a multiple of 3.</td> </tr> <tr> <td><i>c</i></td> <td>Color used in the drawing.</td> </tr> </table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>zbuffer</i>	Pointer to the image that holds the Z buffer.	<i>x</i>	Pointer to array of X coordinates of the points.	<i>y</i>	Pointer to array of Y coordinates of the points.	<i>z</i>	Pointer to array of Z coordinates of the points.	<i>npoints</i>	Number of points in the arrays. npoints must be a multiple of 3.	<i>c</i>	Color used in the drawing.
<i>buffer</i>	Pointer to the image into which the function is drawing.														
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.														
<i>x</i>	Pointer to array of X coordinates of the points.														
<i>y</i>	Pointer to array of Y coordinates of the points.														
<i>z</i>	Pointer to array of Z coordinates of the points.														
<i>npoints</i>	Number of points in the arrays. npoints must be a multiple of 3.														
<i>c</i>	Color used in the drawing.														
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.														
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1" style="margin-left: 20px; width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe								
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														
MT-Level	MT-Safe														
SEE ALSO	mllib_GraphicsDrawTriangleSet_8(3MLIB), mllib_GraphicsDrawTriangleSet_A_8(3MLIB), mllib_GraphicsDrawTriangleSet_AG_8(3MLIB), mllib_GraphicsDrawTriangleSet_AGZ_8(3MLIB), mllib_GraphicsDrawTriangleSet_G_8(3MLIB), mllib_GraphicsDrawTriangleSet_GZ_8(3MLIB), mllib_GraphicsDrawTriangleSet_X_8(3MLIB), mllib_GraphicsDrawTriangleSet_Z_8(3MLIB), attributes(5)														

mllib_GraphicsDrawTriangleSet_G_8(3MLIB)

NAME mllib_GraphicsDrawTriangleSet_G_8, mllib_GraphicsDrawTriangleSet_G_32 – draw triangle set with Gouraud shading where each member can have different vertices

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawTriangleSet_G_8(mllib_image *buffer,
        const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, const
        mllib_s32 *c);

mllib_status mllib_GraphicsDrawTriangleSet_G_32(mllib_image *buffer,
        const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, const
        mllib_s32 *c);
```

DESCRIPTION Each of these functions draws set of triangles with vertices at {(x1,y1),(x2,y2), (x3,y3)}, {(x4,y4),(x5,y5),(x6,y6)}, ..., and {(xn-2,yn-2),(xn-1,yn-1),(xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

x Pointer to array of X coordinates of the points.

y Pointer to array of Y coordinates of the points.

npoints Number of points in the arrays. *npoints* must be a multiple of 3.

c Pointer to array of colors of the points.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawTriangleSet_8(3MLIB), mllib_GraphicsDrawTriangleSet_A_8(3MLIB), mllib_GraphicsDrawTriangleSet_AG_8(3MLIB), mllib_GraphicsDrawTriangleSet_AGZ_8(3MLIB), mllib_GraphicsDrawTriangleSet_AZ_8(3MLIB), mllib_GraphicsDrawTriangleSet_GZ_8(3MLIB), mllib_GraphicsDrawTriangleSet_X_8(3MLIB), mllib_GraphicsDrawTriangleSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawTriangleSet_GZ_8(3MLIB)

NAME	mllib_GraphicsDrawTriangleSet_GZ_8, mllib_GraphicsDrawTriangleSet_GZ_32 – draw triangle set with Gouraud shading and Z buffering, where each member can have different vertices						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawTriangleSet_GZ_8(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, const mllib_s16 *<i>z</i>, mllib_s32 <i>npoints</i>, const mllib_s32 *<i>c</i>); mllib_status mllib_GraphicsDrawTriangleSet_GZ_32(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, const mllib_s16 *<i>z</i>, mllib_s32 <i>npoints</i>, const mllib_s32 *<i>c</i>);</pre>						
DESCRIPTION	Each of these functions draws a set of triangles with vertices at {(x1,y1),(x2,y2), (x3,y3)}, {(x4,y4),(x5,y5),(x6,y6)}, ..., and {(xn-2,yn-2),(xn-1,yn-1),(xn,yn)}.						
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <p><i>buffer</i> Pointer to the image into which the function is drawing.</p> <p><i>zbuffer</i> Pointer to the image that holds the Z buffer.</p> <p><i>x</i> Pointer to array of X coordinates of the points.</p> <p><i>y</i> Pointer to array of Y coordinates of the points.</p> <p><i>z</i> Pointer to array of Z coordinates of the points.</p> <p><i>npoints</i> Number of points in the arrays. npoints must be a multiple of 3.</p> <p><i>c</i> Pointer to array of colors of the points.</p>						
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<p>mllib_GraphicsDrawTriangleSet_8(3MLIB), mllib_GraphicsDrawTriangleSet_A_8(3MLIB), mllib_GraphicsDrawTriangleSet_AG_8(3MLIB), mllib_GraphicsDrawTriangleSet_AGZ_8(3MLIB), mllib_GraphicsDrawTriangleSet_AZ_8(3MLIB), mllib_GraphicsDrawTriangleSet_G_8(3MLIB), mllib_GraphicsDrawTriangleSet_X_8(3MLIB), mllib_GraphicsDrawTriangleSet_Z_8(3MLIB), attributes(5)</p>						

mllib_GraphicsDrawTriangleSet_X_8(3MLIB)

NAME mllib_GraphicsDrawTriangleSet_X_8, mllib_GraphicsDrawTriangleSet_X_32 – draw triangle set in Xor mode where each member can have different vertices

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawTriangleSet_X_8(mllib_image *buffer,
        const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
        mllib_s32 c, mllib_s32 c2);

mllib_status mllib_GraphicsDrawTriangleSet_X_32(mllib_image *buffer,
        const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
        mllib_s32 c, mllib_s32 c2);
```

DESCRIPTION Each of these functions draws set of triangles with vertices at {(x1,y1),(x2,y2), (x3,y3)}, {(x4,y4),(x5,y5),(x6,y6)}, ..., and {(xn-2,yn-2),(xn-1,yn-1),(xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

x Pointer to array of X coordinates of the points.

y Pointer to array of Y coordinates of the points.

npoints Number of points in the arrays. npoints must be a multiple of 3.

c Color used in the drawing.

c2 Alternation color.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawTriangleSet_8(3MLIB), mllib_GraphicsDrawTriangleSet_A_8(3MLIB), mllib_GraphicsDrawTriangleSet_AG_8(3MLIB), mllib_GraphicsDrawTriangleSet_AGZ_8(3MLIB), mllib_GraphicsDrawTriangleSet_AZ_8(3MLIB), mllib_GraphicsDrawTriangleSet_G_8(3MLIB), mllib_GraphicsDrawTriangleSet_GZ_8(3MLIB), mllib_GraphicsDrawTriangleSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawTriangleSet_Z_8(3MLIB)

NAME	mllib_GraphicsDrawTriangleSet_Z_8, mllib_GraphicsDrawTriangleSet_Z_32 – draw triangle set with Z buffering						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawTriangleSet_Z_8(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, const mllib_s16 *<i>z</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>); mllib_status mllib_GraphicsDrawTriangleSet_Z_32(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, const mllib_s16 *<i>z</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>);</pre>						
DESCRIPTION	Each of these functions draws set of triangles with vertices at {(x1,y1),(x2,y2), (x3,y3)}, {(x4,y4),(x5,y5),(x6,y6)}, ..., and {(xn-2,yn-2),(xn-1,yn-1),(xn,yn)}.						
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <p><i>buffer</i> Pointer to the image into which the function is drawing.</p> <p><i>zbuffer</i> Pointer to the image that holds the Z buffer.</p> <p><i>x</i> Pointer to array of X coordinates of the points.</p> <p><i>y</i> Pointer to array of Y coordinates of the points.</p> <p><i>z</i> Pointer to array of Z coordinates of the points.</p> <p><i>npoints</i> Number of points in the arrays. npoints must be a multiple of 3.</p> <p><i>c</i> Color used in the drawing.</p>						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<p>mllib_GraphicsDrawTriangleSet_8(3MLIB), mllib_GraphicsDrawTriangleSet_A_8(3MLIB), mllib_GraphicsDrawTriangleSet_AG_8(3MLIB), mllib_GraphicsDrawTriangleSet_AGZ_8(3MLIB), mllib_GraphicsDrawTriangleSet_AZ_8(3MLIB), mllib_GraphicsDrawTriangleSet_G_8(3MLIB), mllib_GraphicsDrawTriangleSet_GZ_8(3MLIB), mllib_GraphicsDrawTriangleSet_X_8(3MLIB), attributes(5)</p>						

mllib_GraphicsDrawTriangleStripSet_8(3MLIB)

NAME mllib_GraphicsDrawTriangleStripSet_8, mllib_GraphicsDrawTriangleStripSet_32 – draw triangle set where the first side of each member is common to the second side of the previous member

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawTriangleStripSet_8(mllib_image
    *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
    mllib_s32 c);

mllib_status mllib_GraphicsDrawTriangleStripSet_32(mllib_image
    *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
    mllib_s32 c);
```

DESCRIPTION Each of these functions draws a set of triangles with vertices at {(x1,y1),(x2,y2),(x3,y3)}, {(x2,y2),(x3,y3),(x4,y4)}, ..., and {(xn-2,yn-2),(xn-1,yn-1),(xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

<i>buffer</i>	Pointer to the image into which the function is drawing.
<i>x</i>	Pointer to array of X coordinates of the points.
<i>y</i>	Pointer to array of Y coordinates of the points.
<i>npoints</i>	Number of points in the arrays.
<i>c</i>	Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawTriangleStripSet_A_8(3MLIB), mllib_GraphicsDrawTriangleStripSet_AG_8(3MLIB), mllib_GraphicsDrawTriangleStripSet_AGZ(3MLIB), mllib_GraphicsDrawTriangleStripSet_AZ_8(3MLIB), mllib_GraphicsDrawTriangleStripSet_G_8(3MLIB), mllib_GraphicsDrawTriangleStripSet_GZ_8(3MLIB), mllib_GraphicsDrawTriangleStripSet_X_8(3MLIB), mllib_GraphicsDrawTriangleStripSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawTriangleStripSet_A_8(3MLIB)

NAME	mllib_GraphicsDrawTriangleStripSet_A_8, mllib_GraphicsDrawTriangleStripSet_A_32 – draw triangle set with antialiasing where the first side of each member is common to the second side of the previous member										
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_GraphicsDrawTriangleStripSet_A_8(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c); mllib_status mllib_GraphicsDrawTriangleStripSet_A_32(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c);</pre>										
DESCRIPTION	Each of these functions draws a set of triangles with vertices at {(x1,y1),(x2,y2),(x3,y3)}, {(x2,y2),(x3,y3),(x4,y4)}, ..., and {(xn-2,yn-2),(xn-1,yn-1),(xn,yn)}.										
PARAMETERS	Each of the functions takes the following arguments: <table border="0" style="margin-left: 20px;"> <tr> <td style="padding-right: 20px;"><i>buffer</i></td> <td>Pointer to the image into which the function is drawing.</td> </tr> <tr> <td style="padding-right: 20px;"><i>x</i></td> <td>Pointer to array of X coordinates of the points.</td> </tr> <tr> <td style="padding-right: 20px;"><i>y</i></td> <td>Pointer to array of Y coordinates of the points.</td> </tr> <tr> <td style="padding-right: 20px;"><i>npoints</i></td> <td>Number of points in the arrays.</td> </tr> <tr> <td style="padding-right: 20px;"><i>c</i></td> <td>Color used in the drawing.</td> </tr> </table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>x</i>	Pointer to array of X coordinates of the points.	<i>y</i>	Pointer to array of Y coordinates of the points.	<i>npoints</i>	Number of points in the arrays.	<i>c</i>	Color used in the drawing.
<i>buffer</i>	Pointer to the image into which the function is drawing.										
<i>x</i>	Pointer to array of X coordinates of the points.										
<i>y</i>	Pointer to array of Y coordinates of the points.										
<i>npoints</i>	Number of points in the arrays.										
<i>c</i>	Color used in the drawing.										
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.										
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1" style="margin-left: 20px; width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe				
ATTRIBUTE TYPE	ATTRIBUTE VALUE										
Interface Stability	Evolving										
MT-Level	MT-Safe										
SEE ALSO	mllib_GraphicsDrawTriangleStripSet_8(3MLIB) , mllib_GraphicsDrawTriangleStripSet_AG_8(3MLIB) , mllib_GraphicsDrawTriangleStripSet_AGZ(3MLIB) , mllib_GraphicsDrawTriangleStripSet_AZ_8(3MLIB) , mllib_GraphicsDrawTriangleStripSet_G_8(3MLIB) , mllib_GraphicsDrawTriangleStripSet_GZ_8(3MLIB) , mllib_GraphicsDrawTriangleStripSet_X_8(3MLIB) , mllib_GraphicsDrawTriangleStripSet_Z_8(3MLIB) , attributes(5)										

mllib_GraphicsDrawTriangleStripSet_AG_8(3MLIB)

NAME mllib_GraphicsDrawTriangleStripSet_AG_8,
mllib_GraphicsDrawTriangleStripSet_AG_32 – draw triangle set with antialiasing and gouraud shading

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawTriangleStripSet_AG_8(mllib_image
    *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
    const mllib_s32 *c);

mllib_status mllib_GraphicsDrawTriangleStripSet_AG_32(mllib_image
    *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
    const mllib_s32 *c);
```

DESCRIPTION Each of these functions draws a set of triangles with vertices at {(x1,y1),(x2,y2),(x3,y3)}, {(x2,y2),(x3,y3),(x4,y4)}, ..., and {(xn-2,yn-2),(xn-1,yn-1),(xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

x Pointer to array of X coordinates of the points.

y Pointer to array of Y coordinates of the points.

npoints Number of points in the arrays.

c Pointer to array of colors of the points.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawTriangleStripSet_8(3MLIB),
mllib_GraphicsDrawTriangleStripSet_A_8(3MLIB),
mllib_GraphicsDrawTriangleStripSet_AGZ(3MLIB),
mllib_GraphicsDrawTriangleStripSet_AZ_8(3MLIB),
mllib_GraphicsDrawTriangleStripSet_G_8(3MLIB),
mllib_GraphicsDrawTriangleStripSet_GZ_8(3MLIB),
mllib_GraphicsDrawTriangleStripSet_X_8(3MLIB),
mllib_GraphicsDrawTriangleStripSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawTriangleStripSet_AGZ(3MLIB)

NAME	<code>mllib_GraphicsDrawTriangleStripSet_AGZ</code> , <code>mllib_GraphicsDrawTriangleStripSet_AGZ_8</code> , <code>mllib_GraphicsDrawTriangleStripSet_AGZ_32</code> – draw triangle set with antialiasing, Gouraud shading, and Z buffering, where the first side of each member is common to the second side of the previous member														
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_GraphicsDrawTriangleStripSet_AGZ_8(mllib_image *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c); mllib_status mllib_GraphicsDrawTriangleStripSet_AGZ_32(mllib_image *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);</pre>														
DESCRIPTION	Each of these functions draws a set of triangles with vertices at $\{(x_1,y_1),(x_2,y_2), (x_3,y_3)\}$, $\{(x_2,y_2),(x_3,y_3),(x_4,y_4)\}$, ..., and $\{(x_{n-2},y_{n-2}),(x_{n-1},y_{n-1}),(x_n,y_n)\}$.														
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>zbuffer</i></td><td>Pointer to the image that holds the Z buffer.</td></tr><tr><td><i>x</i></td><td>Pointer to array of X coordinates of the points.</td></tr><tr><td><i>y</i></td><td>Pointer to array of Y coordinates of the points.</td></tr><tr><td><i>z</i></td><td>Pointer to array of Z coordinates of the points.</td></tr><tr><td><i>npoints</i></td><td>Number of points in the arrays.</td></tr><tr><td><i>c</i></td><td>Pointer to array of colors of the points.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>zbuffer</i>	Pointer to the image that holds the Z buffer.	<i>x</i>	Pointer to array of X coordinates of the points.	<i>y</i>	Pointer to array of Y coordinates of the points.	<i>z</i>	Pointer to array of Z coordinates of the points.	<i>npoints</i>	Number of points in the arrays.	<i>c</i>	Pointer to array of colors of the points.
<i>buffer</i>	Pointer to the image into which the function is drawing.														
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.														
<i>x</i>	Pointer to array of X coordinates of the points.														
<i>y</i>	Pointer to array of Y coordinates of the points.														
<i>z</i>	Pointer to array of Z coordinates of the points.														
<i>npoints</i>	Number of points in the arrays.														
<i>c</i>	Pointer to array of colors of the points.														
RETURN VALUES	Each of the functions returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .														
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe								
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														
MT-Level	MT-Safe														
SEE ALSO	<code>mllib_GraphicsDrawTriangleStripSet_8(3MLIB)</code> , <code>mllib_GraphicsDrawTriangleStripSet_A_8(3MLIB)</code> , <code>mllib_GraphicsDrawTriangleStripSet_AG_8(3MLIB)</code> , <code>mllib_GraphicsDrawTriangleStripSet_AZ_8(3MLIB)</code> ,														

`mllib_GraphicsDrawTriangleStripSet_AGZ(3MLIB)`

```
mllib_GraphicsDrawTriangleStripSet_G_8(3MLIB),  
mllib_GraphicsDrawTriangleStripSet_GZ_8(3MLIB),  
mllib_GraphicsDrawTriangleStripSet_X_8(3MLIB),  
mllib_GraphicsDrawTriangleStripSet_Z_8(3MLIB), attributes(5)
```

mllib_GraphicsDrawTriangleStripSet_AZ_8(3MLIB)

NAME	<code>mllib_GraphicsDrawTriangleStripSet_AZ_8</code> , <code>mllib_GraphicsDrawTriangleStripSet_AZ_32</code> – draw triangle set with antialiasing and Z buffering, where the first side of each member is common to the second side of the previous member														
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsDrawTriangleStripSet_AZ_8(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, const mllib_s16 *<i>z</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>); mllib_status mllib_GraphicsDrawTriangleStripSet_AZ_32(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, const mllib_s16 *<i>z</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>);</pre>														
DESCRIPTION	Each of these functions draws a set of triangles with vertices at $\{(x_1,y_1),(x_2,y_2),(x_3,y_3)\}$, $\{(x_2,y_2),(x_3,y_3),(x_4,y_4)\}$, ..., and $\{(x_{n-2},y_{n-2}),(x_{n-1},y_{n-1}),(x_n,y_n)\}$.														
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>zbuffer</i></td><td>Pointer to the image that holds the Z buffer.</td></tr><tr><td><i>x</i></td><td>Pointer to array of X coordinates of the points.</td></tr><tr><td><i>y</i></td><td>Pointer to array of Y coordinates of the points.</td></tr><tr><td><i>z</i></td><td>Pointer to array of Z coordinates of the points.</td></tr><tr><td><i>npoints</i></td><td>Number of points in the arrays.</td></tr><tr><td><i>c</i></td><td>Color used in the drawing.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>zbuffer</i>	Pointer to the image that holds the Z buffer.	<i>x</i>	Pointer to array of X coordinates of the points.	<i>y</i>	Pointer to array of Y coordinates of the points.	<i>z</i>	Pointer to array of Z coordinates of the points.	<i>npoints</i>	Number of points in the arrays.	<i>c</i>	Color used in the drawing.
<i>buffer</i>	Pointer to the image into which the function is drawing.														
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.														
<i>x</i>	Pointer to array of X coordinates of the points.														
<i>y</i>	Pointer to array of Y coordinates of the points.														
<i>z</i>	Pointer to array of Z coordinates of the points.														
<i>npoints</i>	Number of points in the arrays.														
<i>c</i>	Color used in the drawing.														
RETURN VALUES	Each of the functions returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .														
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe								
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														
MT-Level	MT-Safe														
SEE ALSO	<code>mllib_GraphicsDrawTriangleStripSet_8(3MLIB)</code> , <code>mllib_GraphicsDrawTriangleStripSet_A_8(3MLIB)</code> , <code>mllib_GraphicsDrawTriangleStripSet_AG_8(3MLIB)</code> , <code>mllib_GraphicsDrawTriangleStripSet_AGZ(3MLIB)</code> , <code>mllib_GraphicsDrawTriangleStripSet_G_8(3MLIB)</code> , <code>mllib_GraphicsDrawTriangleStripSet_GZ_8(3MLIB)</code> , <code>mllib_GraphicsDrawTriangleStripSet_X_8(3MLIB)</code> , <code>mllib_GraphicsDrawTriangleStripSet_Z_8(3MLIB)</code> , <code>attributes(5)</code>														

mllib_GraphicsDrawTriangleStripSet_G_8(3MLIB)

NAME mllib_GraphicsDrawTriangleStripSet_G_8, mllib_GraphicsDrawTriangleStripSet_G_32 – draw triangle set with Gouraud shading where the first side of each member is common to the second side of the previous member

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawTriangleStripSet_G_8(mllib_image
    *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
    const mllib_s32 *c);

mllib_status mllib_GraphicsDrawTriangleStripSet_G_32(mllib_image
    *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
    const mllib_s32 *c);
```

DESCRIPTION Each of these functions draws a set of triangles with vertices at {(x1,y1),(x2,y2),(x3,y3)}, {(x2,y2),(x3,y3),(x4,y4)}, ..., and {(xn-2,yn-2),(xn-1,yn-1),(xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.
x Pointer to array of X coordinates of the points.
y Pointer to array of Y coordinates of the points.
npoints Number of points in the arrays.
c Pointer to array of colors of the points.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawTriangleStripSet_8(3MLIB), mllib_GraphicsDrawTriangleStripSet_A_8(3MLIB), mllib_GraphicsDrawTriangleStripSet_AG_8(3MLIB), mllib_GraphicsDrawTriangleStripSet_AGZ(3MLIB), mllib_GraphicsDrawTriangleStripSet_AZ_8(3MLIB), mllib_GraphicsDrawTriangleStripSet_GZ_8(3MLIB), mllib_GraphicsDrawTriangleStripSet_X_8(3MLIB), mllib_GraphicsDrawTriangleStripSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawTriangleStripSet_GZ_8(3MLIB)

NAME mllib_GraphicsDrawTriangleStripSet_GZ_8,
mllib_GraphicsDrawTriangleStripSet_GZ_32 – draw triangle set with Gouraud shading and Z buffering, where the first side of each member is common to the second side of the previous member

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
mllib_status mllib_GraphicsDrawTriangleStripSet_GZ_8(mllib_image  
    *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16  
    *y, const mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);  
  
mllib_status mllib_GraphicsDrawTriangleStripSet_GZ_32(mllib_image  
    *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16  
    *y, const mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);
```

DESCRIPTION Each of these functions draws a set of triangles with vertices at {(x1,y1),(x2,y2), (x3,y3)}, {(x2,y2),(x3,y3),(x4,y4)}, ..., and {(xn-2,yn-2),(xn-1,yn-1),(xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

<i>buffer</i>	Pointer to the image into which the function is drawing.
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.
<i>x</i>	Pointer to array of X coordinates of the points.
<i>y</i>	Pointer to array of Y coordinates of the points.
<i>z</i>	Pointer to array of Z coordinates of the points.
<i>npoints</i>	Number of points in the arrays.
<i>c</i>	Pointer to array of colors of the points.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawTriangleStripSet_8(3MLIB),
mllib_GraphicsDrawTriangleStripSet_A_8(3MLIB),
mllib_GraphicsDrawTriangleStripSet_AG_8(3MLIB),
mllib_GraphicsDrawTriangleStripSet_AGZ(3MLIB),
mllib_GraphicsDrawTriangleStripSet_AZ_8(3MLIB),
mllib_GraphicsDrawTriangleStripSet_G_8(3MLIB),
mllib_GraphicsDrawTriangleStripSet_X_8(3MLIB),
mllib_GraphicsDrawTriangleStripSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawTriangleStripSet_X_8(3MLIB)

NAME mllib_GraphicsDrawTriangleStripSet_X_8, mllib_GraphicsDrawTriangleStripSet_X_32 – draw triangle set in Xor mode where the first side of each member is common to the second side of the previous member

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawTriangleStripSet_X_8(mllib_image
    *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
    mllib_s32 c, mllib_s32 c2);

mllib_status mllib_GraphicsDrawTriangleStripSet_X_32(mllib_image
    *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
    mllib_s32 c, mllib_s32 c2);
```

DESCRIPTION Each of these functions draws a set of triangles with vertices at {(x1,y1),(x2,y2),(x3,y3)}, {(x2,y2),(x3,y3),(x4,y4)}, ..., and {(xn-2,yn-2),(xn-1,yn-1),(xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.
x Pointer to array of X coordinates of the points.
y Pointer to array of Y coordinates of the points.
npoints Number of points in the arrays.
c Color used in the drawing.
c2 Alternation color.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawTriangleStripSet_8(3MLIB), mllib_GraphicsDrawTriangleStripSet_A_8(3MLIB), mllib_GraphicsDrawTriangleStripSet_AG_8(3MLIB), mllib_GraphicsDrawTriangleStripSet_AGZ(3MLIB), mllib_GraphicsDrawTriangleStripSet_AZ_8(3MLIB), mllib_GraphicsDrawTriangleStripSet_G_8(3MLIB), mllib_GraphicsDrawTriangleStripSet_GZ_8(3MLIB), mllib_GraphicsDrawTriangleStripSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsDrawTriangleStripSet_Z_8(3MLIB)

NAME	mllib_GraphicsDrawTriangleStripSet_Z_8, mllib_GraphicsDrawTriangleStripSet_Z_32 – draw triangle set with Z buffering where the first side of each member is common to the second side of the previous member														
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_GraphicsDrawTriangleStripSet_Z_8(mllib_image *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *z, mllib_s32 npoints, mllib_s32 c); mllib_status mllib_GraphicsDrawTriangleStripSet_Z_32(mllib_image *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *z, mllib_s32 npoints, mllib_s32 c);</pre>														
DESCRIPTION	Each of these functions draws a set of triangles with vertices at {(x1,y1),(x2,y2), (x3,y3)}, {(x2,y2),(x3,y3),(x4,y4)}, ..., and {(xn-2,yn-2),(xn-1,yn-1),(xn,yn)}.														
PARAMETERS	Each of the functions takes the following arguments: <table border="0" style="margin-left: 20px;"> <tr> <td style="padding-right: 20px;"><i>buffer</i></td> <td>Pointer to the image into which the function is drawing.</td> </tr> <tr> <td style="padding-right: 20px;"><i>zbuffer</i></td> <td>Pointer to the image that holds the Z buffer.</td> </tr> <tr> <td style="padding-right: 20px;"><i>x</i></td> <td>Pointer to array of X coordinates of the points.</td> </tr> <tr> <td style="padding-right: 20px;"><i>y</i></td> <td>Pointer to array of Y coordinates of the points.</td> </tr> <tr> <td style="padding-right: 20px;"><i>z</i></td> <td>Pointer to array of Z coordinates of the points.</td> </tr> <tr> <td style="padding-right: 20px;"><i>npoints</i></td> <td>Number of points in the arrays.</td> </tr> <tr> <td style="padding-right: 20px;"><i>c</i></td> <td>Color used in the drawing.</td> </tr> </table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>zbuffer</i>	Pointer to the image that holds the Z buffer.	<i>x</i>	Pointer to array of X coordinates of the points.	<i>y</i>	Pointer to array of Y coordinates of the points.	<i>z</i>	Pointer to array of Z coordinates of the points.	<i>npoints</i>	Number of points in the arrays.	<i>c</i>	Color used in the drawing.
<i>buffer</i>	Pointer to the image into which the function is drawing.														
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.														
<i>x</i>	Pointer to array of X coordinates of the points.														
<i>y</i>	Pointer to array of Y coordinates of the points.														
<i>z</i>	Pointer to array of Z coordinates of the points.														
<i>npoints</i>	Number of points in the arrays.														
<i>c</i>	Color used in the drawing.														
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.														
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1" style="margin-left: 20px; width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe								
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														
MT-Level	MT-Safe														
SEE ALSO	mllib_GraphicsDrawTriangleStripSet_8(3MLIB), mllib_GraphicsDrawTriangleStripSet_A_8(3MLIB), mllib_GraphicsDrawTriangleStripSet_AG_8(3MLIB), mllib_GraphicsDrawTriangleStripSet_AGZ(3MLIB), mllib_GraphicsDrawTriangleStripSet_AZ_8(3MLIB), mllib_GraphicsDrawTriangleStripSet_G_8(3MLIB), mllib_GraphicsDrawTriangleStripSet_GZ_8(3MLIB), mllib_GraphicsDrawTriangleStripSet_X_8(3MLIB), attributes(5)														

mllib_GraphicsDrawTriangle_X_8(3MLIB)

NAME mllib_GraphicsDrawTriangle_X_8, mllib_GraphicsDrawTriangle_X_32 – draw triangle in Xor mode

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawTriangle_X_8(mllib_image *buffer,
      mllib_s16 x1, mllib_s16 y1, mllib_s16 x2, mllib_s16 y2, mllib_s16
      x3, mllib_s16 y3, mllib_s32 c, mllib_s32 c2);

mllib_status mllib_GraphicsDrawTriangle_X_32(mllib_image *buffer,
      mllib_s16 x1, mllib_s16 y1, mllib_s16 x2, mllib_s16 y2, mllib_s16
      x3, mllib_s16 y3, mllib_s32 c, mllib_s32 c2);
```

DESCRIPTION Each of these functions draws a triangle with the vertices at (x1,y1), (x2,y2), and (x3,y3).

PARAMETERS Each of the functions takes the following arguments:

<i>buffer</i>	Pointer to the image into which the function is drawing.
<i>x1</i>	X coordinate of the first vertex.
<i>y1</i>	Y coordinate of the first vertex.
<i>x2</i>	X coordinate of the second vertex.
<i>y2</i>	Y coordinate of the second vertex.
<i>x3</i>	X coordinate of the third vertex.
<i>y3</i>	Y coordinate of the third vertex.
<i>c</i>	Color used in the drawing.
<i>c2</i>	Alternation color.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsDrawTriangle_8(3MLIB),
mllib_GraphicsDrawTriangle_A_8(3MLIB),
mllib_GraphicsDrawTriangle_AG_8(3MLIB),
mllib_GraphicsDrawTriangle_AGZ_8(3MLIB),

`mllib_GraphicsDrawTriangle_X_8(3MLIB)`

```
mllib_GraphicsDrawTriangle_AZ_8(3MLIB),  
mllib_GraphicsDrawTriangle_G_8(3MLIB),  
mllib_GraphicsDrawTriangle_GZ_8(3MLIB),  
mllib_GraphicsDrawTriangle_Z_8(3MLIB), attributes(5)
```

mllib_GraphicsDrawTriangle_Z_8(3MLIB)

NAME mllib_GraphicsDrawTriangle_Z_8, mllib_GraphicsDrawTriangle_Z_32 – draw triangle with Z buffering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsDrawTriangle_Z_8(mllib_image *buffer,
      mllib_image *zbuffer, mllib_s16 x1, mllib_s16 y1, mllib_s16 z1,
      mllib_s16 x2, mllib_s16 y2, mllib_s16 z2, mllib_s16 x3, mllib_s16
      y3, mllib_s16 z3, mllib_s32 c);

mllib_status mllib_GraphicsDrawTriangle_Z_32(mllib_image *buffer,
      mllib_image *zbuffer, mllib_s16 x1, mllib_s16 y1, mllib_s16 z1,
      mllib_s16 x2, mllib_s16 y2, mllib_s16 z2, mllib_s16 x3, mllib_s16
      y3, mllib_s16 z3, mllib_s32 c);
```

DESCRIPTION Each of these functions draws a triangle with the vertices at (x1,y1), (x2,y2), and (x3,y3).

PARAMETERS Each of the functions takes the following arguments:

<i>buffer</i>	Pointer to the image into which the function is drawing.
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.
<i>x1</i>	X coordinate of the first vertex.
<i>y1</i>	Y coordinate of the first vertex.
<i>z1</i>	Z coordinate of the first vertex.
<i>x2</i>	X coordinate of the second vertex.
<i>y2</i>	Y coordinate of the second vertex.
<i>z2</i>	Z coordinate of the second vertex.
<i>x3</i>	X coordinate of the third vertex.
<i>y3</i>	Y coordinate of the third vertex.
<i>z3</i>	Z coordinate of the third vertex.
<i>c</i>	Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

mllib_GraphicsDrawTriangle_Z_8(3MLIB)

SEE ALSO | mllib_GraphicsDrawTriangle_8(3MLIB),
mllib_GraphicsDrawTriangle_A_8(3MLIB),
mllib_GraphicsDrawTriangle_AG_8(3MLIB),
mllib_GraphicsDrawTriangle_AGZ_8(3MLIB),
mllib_GraphicsDrawTriangle_AZ_8(3MLIB),
mllib_GraphicsDrawTriangle_G_8(3MLIB),
mllib_GraphicsDrawTriangle_GZ_8(3MLIB),
mllib_GraphicsDrawTriangle_X_8(3MLIB), attributes(5)

NAME mllib_GraphicsFillArc_8, mllib_GraphicsFillArc_32 – draw filled arc

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillArc_8(mllib_image *buffer, mllib_s16 x,
    mllib_s16 y, mllib_s32 r, mllib_f32 t1, mllib_f32 t2, mllib_s32 c);

mllib_status mllib_GraphicsFillArc_32(mllib_image *buffer, mllib_s16 x,
    mllib_s16 y, mllib_s32 r, mllib_f32 t1, mllib_f32 t2, mllib_s32 c);
```

DESCRIPTION Each of these functions draws a filled arc with the center at (x, y) , radius r , start angle $t1$, and end angle $t2$.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

x X coordinate of the center.

y Y coordinate of the center.

r Radius of the arc.

t1 Start angle of the arc in radians.

t2 End angle of the arc in radians.

c Color used in the drawing.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_GraphicsFillArc_A_8(3MLIB)`, `mllib_GraphicsFillArc_X_8(3MLIB)`, `attributes(5)`

mllib_GraphicsFillArc_A_8(3MLIB)

NAME mllib_GraphicsFillArc_A_8, mllib_GraphicsFillArc_A_32 – draw filled arc with antialiasing

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillArc_A_8(mllib_image *buffer, mllib_s16
    x, mllib_s16 y, mllib_s32 r, mllib_f32 t1, mllib_f32 t2, mllib_s32
    c);

mllib_status mllib_GraphicsFillArc_A_32(mllib_image *buffer, mllib_s16
    x, mllib_s16 y, mllib_s32 r, mllib_f32 t1, mllib_f32 t2, mllib_s32
    c);
```

DESCRIPTION Each of these functions draws a filled arc with the center at (*x*, *y*), radius *r*, start angle *t1*, and end angle *t2*.

PARAMETERS Each of the functions takes the following arguments:

<i>buffer</i>	Pointer to the image into which the function is drawing.
<i>x</i>	X coordinate of the center.
<i>y</i>	Y coordinate of the center.
<i>r</i>	Radius of the arc.
<i>t1</i>	Start angle of the arc in radians.
<i>t2</i>	End angle of the arc in radians.
<i>c</i>	Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_GraphicsFillArc_8\(3MLIB\)](#), [mllib_GraphicsFillArc_X_8\(3MLIB\)](#), [attributes\(5\)](#)

NAME mllib_GraphicsFillArc_X_8, mllib_GraphicsFillArc_X_32 – draw filled arc in Xor mode

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillArc_X_8(mllib_image *buffer, mllib_s16
    x, mllib_s16 y, mllib_s32 r, mllib_f32 t1, mllib_f32 t2, mllib_s32 c,
    mllib_s32 c2);

mllib_status mllib_GraphicsFillArc_X_32(mllib_image *buffer, mllib_s16
    x, mllib_s16 y, mllib_s32 r, mllib_f32 t1, mllib_f32 t2, mllib_s32 c,
    mllib_s32 c2);
```

DESCRIPTION Each of these functions draws a filled arc with the center at (x,y), radius r, start angle q1, and end angle q2.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

x X coordinate of the center.

y Y coordinate of the center.

r Radius of the arc.

t1 Start angle of the arc in radians.

t2 End angle of the arc in radians.

c Color used in the drawing.

c2 Alternation color.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsFillArc_8(3MLIB), mllib_GraphicsFillArc_A_8(3MLIB), attributes(5)

mllib_GraphicsFillCircle_8(3MLIB)

NAME mllib_GraphicsFillCircle_8, mllib_GraphicsFillCircle_32 – draw filled circle

SYNOPSIS cc [*flag...*] *file...* -lmllib [*library...*]
#include <mllib.h>

```
mllib_status mllib_GraphicsFillCircle_8(mllib_image *buffer, mllib_s16
    x, mllib_s16 y, mllib_s32 r, mllib_s32 c);
mllib_status mllib_GraphicsFillCircle_32(mllib_image *buffer, mllib_s16
    x, mllib_s16 y, mllib_s32 r, mllib_s32 c);
```

DESCRIPTION Each of these functions draws a filled circle with the center at (x,y) and radius r.

PARAMETERS Each of the functions takes the following arguments:

<i>buffer</i>	Pointer to the image into which the function is drawing.
<i>x</i>	X coordinate of the center.
<i>y</i>	Y coordinate of the center.
<i>r</i>	Radius of the arc.
<i>c</i>	Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_GraphicsFillCircle_A_8\(3MLIB\)](#),
[mllib_GraphicsFillCircle_X_8\(3MLIB\)](#), [attributes\(5\)](#)

mllib_GraphicsFillCircle_A_8(3MLIB)

NAME mllib_GraphicsFillCircle_A_8, mllib_GraphicsFillCircle_A_32 – draw filled circle with antialiasing

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillCircle_A_8(mllib_image *buffer,
      mllib_s16 x, mllib_s16 y, mllib_s32 r, mllib_s32 c);

mllib_status mllib_GraphicsFillCircle_A_32(mllib_image *buffer,
      mllib_s16 x, mllib_s16 y, mllib_s32 r, mllib_s32 c);
```

DESCRIPTION Each of these functions draws a filled circle with the center at (x,y) and radius r.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.
x X coordinate of the center.
y Y coordinate of the center.
r Radius of the arc.
c Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsFillCircle_8(3MLIB),
 mllib_GraphicsFillCircle_X_8(3MLIB), attributes(5)

mllib_GraphicsFillCircle_X_8(3MLIB)

NAME mllib_GraphicsFillCircle_X_8, mllib_GraphicsFillCircle_X_32 – draw filled circle in Xor mode

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillCircle_X_8(mllib_image *buffer,
      mllib_s16 x, mllib_s16 y, mllib_s32 r, mllib_s32 c, mllib_s32 c2);
mllib_status mllib_GraphicsFillCircle_X_32(mllib_image *buffer,
      mllib_s16 x, mllib_s16 y, mllib_s32 r, mllib_s32 c, mllib_s32 c2);
```

DESCRIPTION Each of these functions draws a filled circle with the center at (x,y) and radius r.

PARAMETERS Each of the functions takes the following arguments:

<i>buffer</i>	Pointer to the image into which the function is drawing.
<i>x</i>	X coordinate of the center.
<i>y</i>	Y coordinate of the center.
<i>r</i>	Radius of the arc.
<i>c</i>	Color used in the drawing.
<i>c2</i>	Alternation color.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_GraphicsFillCircle_8\(3MLIB\)](#),
[mllib_GraphicsFillCircle_A_8\(3MLIB\)](#), [attributes\(5\)](#)

mllib_GraphicsFillEllipse_8(3MLIB)

NAME mllib_GraphicsFillEllipse_8, mllib_GraphicsFillEllipse_32 – draw filled ellipse

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillEllipse_8(mllib_image *buffer, mllib_s16
    x, mllib_s16 y, mllib_s32 a, mllib_s32 b, mllib_f32 t, mllib_s32 c);

mllib_status mllib_GraphicsFillEllipse_32(mllib_image *buffer,
    mllib_s16 x, mllib_s16 y, mllib_s32 a, mllib_s32 b, mllib_f32 t,
    mllib_s32 c);
```

DESCRIPTION Each of these functions draws a filled ellipse with the center at (x, y) , major semiaxis a , and minor semiaxis b . The angle of the major semiaxis is t clockwise from the X axis.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

x X coordinate of the center.

y Y coordinate of the center.

a Major semiaxis of the ellipse.

b Minor semiaxis of the ellipse.

t Angle of major semiaxis in radians, clockwise.

c Color used in the drawing.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_GraphicsFillEllipse_A_8\(3MLIB\)](#),
[mllib_GraphicsFillEllipse_X_8\(3MLIB\)](#), `attributes(5)`

mllib_GraphicsFillEllipse_A_8(3MLIB)

NAME	mllib_GraphicsFillEllipse_A_8, mllib_GraphicsFillEllipse_A_32 – draw filled ellipse with antialiasing														
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsFillEllipse_A_8(mllib_image *<i>buffer</i>, mllib_s16 <i>x</i>, mllib_s16 <i>y</i>, mllib_s32 <i>a</i>, mllib_s32 <i>b</i>, mllib_f32 <i>t</i>, mllib_s32 <i>c</i>); mllib_status mllib_GraphicsFillEllipse_A_32(mllib_image *<i>buffer</i>, mllib_s16 <i>x</i>, mllib_s16 <i>y</i>, mllib_s32 <i>a</i>, mllib_s32 <i>b</i>, mllib_f32 <i>t</i>, mllib_s32 <i>c</i>);</pre>														
DESCRIPTION	Each of these functions draws a filled ellipse with the center at (<i>x</i> , <i>y</i>), major semiaxis <i>a</i> , and minor semiaxis <i>b</i> . The angle of the major semiaxis is <i>t</i> clockwise from the X axis.														
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>x</i></td><td>X coordinate of the center.</td></tr><tr><td><i>y</i></td><td>Y coordinate of the center.</td></tr><tr><td><i>a</i></td><td>Major semiaxis of the ellipse.</td></tr><tr><td><i>b</i></td><td>Minor semiaxis of the ellipse.</td></tr><tr><td><i>t</i></td><td>Angle of major semiaxis in radians, clockwise.</td></tr><tr><td><i>c</i></td><td>Color used in the drawing.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>x</i>	X coordinate of the center.	<i>y</i>	Y coordinate of the center.	<i>a</i>	Major semiaxis of the ellipse.	<i>b</i>	Minor semiaxis of the ellipse.	<i>t</i>	Angle of major semiaxis in radians, clockwise.	<i>c</i>	Color used in the drawing.
<i>buffer</i>	Pointer to the image into which the function is drawing.														
<i>x</i>	X coordinate of the center.														
<i>y</i>	Y coordinate of the center.														
<i>a</i>	Major semiaxis of the ellipse.														
<i>b</i>	Minor semiaxis of the ellipse.														
<i>t</i>	Angle of major semiaxis in radians, clockwise.														
<i>c</i>	Color used in the drawing.														
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.														
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe								
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														
MT-Level	MT-Safe														
SEE ALSO	<code>mllib_GraphicsFillEllipse_8(3MLIB)</code> , <code>mllib_GraphicsFillEllipse_x_8(3MLIB)</code> , <code>attributes(5)</code>														

NAME mllib_GraphicsFillEllipse_X_8, mllib_GraphicsFillEllipse_X_32 – draw filled ellipse in Xor mode

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillEllipse_X_8(mllib_image *buffer,
      mllib_s16 x, mllib_s16 y, mllib_s32 a, mllib_s32 b, mllib_f32 t,
      mllib_s32 c, mllib_s32 c2);

mllib_status mllib_GraphicsFillEllipse_X_32(mllib_image *buffer,
      mllib_s16 x, mllib_s16 y, mllib_s32 a, mllib_s32 b, mllib_f32 t,
      mllib_s32 c, mllib_s32 c2);
```

DESCRIPTION Each of these functions draws a filled ellipse with the center at (*x*, *y*), major semiaxis *a*, and minor semiaxis *b*. The angle of the major semiaxis is *t* clockwise from the X axis.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

x X coordinate of the center.

y Y coordinate of the center.

a Major semiaxis of the ellipse.

b Minor semiaxis of the ellipse.

t Angle of major semiaxis in radians, clockwise.

c Color used in the drawing.

c2 Alternation color.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsFillEllipse_8(3MLIB), mllib_GraphicsFillEllipse_A_8(3MLIB), attributes(5)

mllib_GraphicsFillPolygon_8(3MLIB)

NAME	mllib_GraphicsFillPolygon_8, mllib_GraphicsFillPolygon_32 – draw filled polygon						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsFillPolygon_8(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>); mllib_status mllib_GraphicsFillPolygon_32(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>);</pre>						
DESCRIPTION	Each of these functions draws a filled polygon enclosing (x1,y1), (x2,y2), ..., and (xn,yn).						
PARAMETERS	Each of the functions takes the following arguments: <i>buffer</i> Pointer to the image into which the function is drawing. <i>x</i> Pointer to the array of X coordinates of the vertices. <i>y</i> Pointer to the array of Y coordinates of the vertices. <i>npoints</i> Number of vertices in the arrays. <i>c</i> Color used in the drawing.						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_GraphicsFillPolygon_A_8(3MLIB), mllib_GraphicsFillPolygon_AG_8(3MLIB), mllib_GraphicsFillPolygon_AGZ_8(3MLIB), mllib_GraphicsFillPolygon_AZ_8(3MLIB), mllib_GraphicsFillPolygon_G_8(3MLIB), mllib_GraphicsFillPolygon_GZ_8(3MLIB), mllib_GraphicsFillPolygon_X_8(3MLIB), mllib_GraphicsFillPolygon_Z_8(3MLIB), attributes(5)						

mllib_GraphicsFillPolygon_A_8(3MLIB)

NAME mllib_GraphicsFillPolygon_A_8, mllib_GraphicsFillPolygon_A_32 – draw filled polygon with antialiasing

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillPolygon_A_8(mllib_image *buffer, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c);

mllib_status mllib_GraphicsFillPolygon_A_32(mllib_image *buffer, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c);
```

DESCRIPTION Each of these functions draws filled polygon enclosing (x1,y1), (x2,y2), ..., and (xn,yn).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.
x Pointer to the array of X coordinates of the vertices.
y Pointer to the array of Y coordinates of the vertices.
npoints Number of vertices in the arrays.
c Color used in the drawing.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsFillPolygon_8(3MLIB),
 mllib_GraphicsFillPolygon_AG_8(3MLIB),
 mllib_GraphicsFillPolygon_AGZ_8(3MLIB),
 mllib_GraphicsFillPolygon_AZ_8(3MLIB),
 mllib_GraphicsFillPolygon_G_8(3MLIB),
 mllib_GraphicsFillPolygon_GZ_8(3MLIB),
 mllib_GraphicsFillPolygon_X_8(3MLIB),
 mllib_GraphicsFillPolygon_Z_8(3MLIB), attributes(5)

mllib_GraphicsFillPolygon_AG_8(3MLIB)

NAME	mllib_GraphicsFillPolygon_AG_8, mllib_GraphicsFillPolygon_AG_32 – draw filled polygon with antialiasing and Gouraud shading										
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsFillPolygon_AG_8(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, const mllib_s32 *<i>c</i>); mllib_status mllib_GraphicsFillPolygon_AG_32(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, const mllib_s32 *<i>c</i>);</pre>										
DESCRIPTION	Each of these functions draws a filled polygon enclosing (x1,y1), (x2,y2), ..., and (xn,yn).										
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>x</i></td><td>Pointer to the array of X coordinates of the vertices.</td></tr><tr><td><i>y</i></td><td>Pointer to the array of Y coordinates of the vertices.</td></tr><tr><td><i>npoints</i></td><td>Number of vertices in the arrays.</td></tr><tr><td><i>c</i></td><td>Pointer to the array of color of the vertices.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>x</i>	Pointer to the array of X coordinates of the vertices.	<i>y</i>	Pointer to the array of Y coordinates of the vertices.	<i>npoints</i>	Number of vertices in the arrays.	<i>c</i>	Pointer to the array of color of the vertices.
<i>buffer</i>	Pointer to the image into which the function is drawing.										
<i>x</i>	Pointer to the array of X coordinates of the vertices.										
<i>y</i>	Pointer to the array of Y coordinates of the vertices.										
<i>npoints</i>	Number of vertices in the arrays.										
<i>c</i>	Pointer to the array of color of the vertices.										
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.										
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe				
ATTRIBUTE TYPE	ATTRIBUTE VALUE										
Interface Stability	Evolving										
MT-Level	MT-Safe										
SEE ALSO	mllib_GraphicsFillPolygon_8(3MLIB), mllib_GraphicsFillPolygon_A_8(3MLIB), mllib_GraphicsFillPolygon_AGZ_8(3MLIB), mllib_GraphicsFillPolygon_AZ_8(3MLIB), mllib_GraphicsFillPolygon_G_8(3MLIB), mllib_GraphicsFillPolygon_GZ_8(3MLIB), mllib_GraphicsFillPolygon_X_8(3MLIB), mllib_GraphicsFillPolygon_Z_8(3MLIB), attributes(5)										

mllib_GraphicsFillPolygon_AGZ_8(3MLIB)

NAME mllib_GraphicsFillPolygon_AGZ_8, mllib_GraphicsFillPolygon_AGZ_32 – draw filled polygon with antialiasing, Gouraud shading, and Z buffering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillPolygon_AGZ_8(mllib_image *buffer,
        mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const
        mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);

mllib_status mllib_GraphicsFillPolygon_AGZ_32(mllib_image *buffer,
        mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const
        mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);
```

DESCRIPTION Each of these functions draws a polygon enclosing (x1,y1), (x2,y2), ..., and (xn,yn).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.
zbuffer Pointer to the image that holds the Z buffer.
x Pointer to the array of X coordinates of the vertices.
y Pointer to the array of Y coordinates of the vertices.
z Pointer to the array of Z coordinates of the vertices.
npoints Number of vertices in the arrays.
c Pointer to the array of colors of the vertices.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsFillPolygon_8(3MLIB),
mllib_GraphicsFillPolygon_A_8(3MLIB),
mllib_GraphicsFillPolygon_AG_8(3MLIB),
mllib_GraphicsFillPolygon_AZ_8(3MLIB),
mllib_GraphicsFillPolygon_G_8(3MLIB),
mllib_GraphicsFillPolygon_GZ_8(3MLIB),
mllib_GraphicsFillPolygon_X_8(3MLIB),
mllib_GraphicsFillPolygon_Z_8(3MLIB), attributes(5)

mllib_GraphicsFillPolygon_AZ_8(3MLIB)

NAME	mllib_GraphicsFillPolygon_AZ_8, mllib_GraphicsFillPolygon_AZ_32 – draw filled polygon with antialiasing and Z buffering														
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsFillPolygon_AZ_8(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, const mllib_s16 *<i>z</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>); mllib_status mllib_GraphicsFillPolygon_AZ_32(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, const mllib_s16 *<i>z</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>);</pre>														
DESCRIPTION	Each of these functions draws a filled polygon enclosing (x1,y1), (x2,y2), ..., and (xn,yn).														
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>zbuffer</i></td><td>Pointer to the image that holds the Z buffer.</td></tr><tr><td><i>x</i></td><td>Pointer to the array of X coordinates of the vertices.</td></tr><tr><td><i>y</i></td><td>Pointer to the array of Y coordinates of the vertices.</td></tr><tr><td><i>z</i></td><td>Pointer to the array of Z coordinates of the vertices.</td></tr><tr><td><i>npoints</i></td><td>Number of vertices in the arrays.</td></tr><tr><td><i>c</i></td><td>Color used in the drawing.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>zbuffer</i>	Pointer to the image that holds the Z buffer.	<i>x</i>	Pointer to the array of X coordinates of the vertices.	<i>y</i>	Pointer to the array of Y coordinates of the vertices.	<i>z</i>	Pointer to the array of Z coordinates of the vertices.	<i>npoints</i>	Number of vertices in the arrays.	<i>c</i>	Color used in the drawing.
<i>buffer</i>	Pointer to the image into which the function is drawing.														
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.														
<i>x</i>	Pointer to the array of X coordinates of the vertices.														
<i>y</i>	Pointer to the array of Y coordinates of the vertices.														
<i>z</i>	Pointer to the array of Z coordinates of the vertices.														
<i>npoints</i>	Number of vertices in the arrays.														
<i>c</i>	Color used in the drawing.														
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.														
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe								
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														
MT-Level	MT-Safe														
SEE ALSO	mllib_GraphicsFillPolygon_8(3MLIB), mllib_GraphicsFillPolygon_A_8(3MLIB), mllib_GraphicsFillPolygon_AG_8(3MLIB), mllib_GraphicsFillPolygon_AGZ_8(3MLIB), mllib_GraphicsFillPolygon_G_8(3MLIB), mllib_GraphicsFillPolygon_GZ_8(3MLIB), mllib_GraphicsFillPolygon_X_8(3MLIB), mllib_GraphicsFillPolygon_Z_8(3MLIB), attributes(5)														

mllib_GraphicsFillPolygon_G_8(3MLIB)

NAME mllib_GraphicsFillPolygon_G_8, mllib_GraphicsFillPolygon_G_32 – draw filled polygon with Gouraud shading

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillPolygon_G_8(mllib_image *buffer, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, const
    mllib_s32 *c);

mllib_status mllib_GraphicsFillPolygon_G_32(mllib_image *buffer, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, const
    mllib_s32 *c);
```

DESCRIPTION Each of these functions draws a filled polygon enclosing (x1,y1), (x2,y2), ..., and (xn,yn).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.
x Pointer to the array of X coordinates of the vertices.
y Pointer to the array of Y coordinates of the vertices.
npoints Number of vertices in the arrays.
c Pointer to the array of color of the vertices.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsFillPolygon_8(3MLIB),
 mllib_GraphicsFillPolygon_A_8(3MLIB),
 mllib_GraphicsFillPolygon_AG_8(3MLIB),
 mllib_GraphicsFillPolygon_AGZ_8(3MLIB),
 mllib_GraphicsFillPolygon_AZ_8(3MLIB),
 mllib_GraphicsFillPolygon_GZ_8(3MLIB),
 mllib_GraphicsFillPolygon_X_8(3MLIB),
 mllib_GraphicsFillPolygon_Z_8(3MLIB), attributes(5)

mllib_GraphicsFillPolygon_GZ_8(3MLIB)

NAME	mllib_GraphicsFillPolygon_GZ_8, mllib_GraphicsFillPolygon_GZ_32 – draw filled polygon with Gouraud shading and Z buffering														
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsFillPolygon_GZ_8(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, const mllib_s16 *<i>z</i>, mllib_s32 <i>npoints</i>, const mllib_s32 *<i>c</i>); mllib_status mllib_GraphicsFillPolygon_GZ_32(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, const mllib_s16 *<i>z</i>, mllib_s32 <i>npoints</i>, const mllib_s32 *<i>c</i>);</pre>														
DESCRIPTION	Each of these functions draws a polygon enclosing (x1,y1), (x2,y2), ..., and (xn,yn).														
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>zbuffer</i></td><td>Pointer to the image that holds the Z buffer.</td></tr><tr><td><i>x</i></td><td>Pointer to the array of X coordinates of the vertices.</td></tr><tr><td><i>y</i></td><td>Pointer to the array of Y coordinates of the vertices.</td></tr><tr><td><i>z</i></td><td>Pointer to the array of Z coordinates of the vertices.</td></tr><tr><td><i>npoints</i></td><td>Number of vertices in the arrays.</td></tr><tr><td><i>c</i></td><td>Pointer to the array of colors of the vertices.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>zbuffer</i>	Pointer to the image that holds the Z buffer.	<i>x</i>	Pointer to the array of X coordinates of the vertices.	<i>y</i>	Pointer to the array of Y coordinates of the vertices.	<i>z</i>	Pointer to the array of Z coordinates of the vertices.	<i>npoints</i>	Number of vertices in the arrays.	<i>c</i>	Pointer to the array of colors of the vertices.
<i>buffer</i>	Pointer to the image into which the function is drawing.														
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.														
<i>x</i>	Pointer to the array of X coordinates of the vertices.														
<i>y</i>	Pointer to the array of Y coordinates of the vertices.														
<i>z</i>	Pointer to the array of Z coordinates of the vertices.														
<i>npoints</i>	Number of vertices in the arrays.														
<i>c</i>	Pointer to the array of colors of the vertices.														
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.														
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe								
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														
MT-Level	MT-Safe														
SEE ALSO	mllib_GraphicsFillPolygon_8(3MLIB), mllib_GraphicsFillPolygon_A_8(3MLIB), mllib_GraphicsFillPolygon_AG_8(3MLIB), mllib_GraphicsFillPolygon_AGZ_8(3MLIB), mllib_GraphicsFillPolygon_AZ_8(3MLIB), mllib_GraphicsFillPolygon_G_8(3MLIB), mllib_GraphicsFillPolygon_X_8(3MLIB), mllib_GraphicsFillPolygon_Z_8(3MLIB), attributes(5)														

mlib_GraphicsFillPolygon_X_8(3MLIB)

NAME mlib_GraphicsFillPolygon_X_8, mlib_GraphicsFillPolygon_X_32 – draw filled polygon in Xor mode

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_GraphicsFillPolygon_X_8(mlib_image *buffer, const
    mlib_s16 *x, const mlib_s16 *y, mlib_s32 npoints, mlib_s32 c,
    mlib_s32 c2);

mlib_status mlib_GraphicsFillPolygon_X_32(mlib_image *buffer, const
    mlib_s16 *x, const mlib_s16 *y, mlib_s32 npoints, mlib_s32 c,
    mlib_s32 c2);
```

DESCRIPTION Each of these functions draws a filled polygon enclosing (x1,y1), (x2,y2), ..., and (xn,yn).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

x Pointer to the array of X coordinates of the vertices.

y Pointer to the array of Y coordinates of the vertices.

npoints Number of vertices in the arrays.

c Color used in the drawing.

c2 Alternation color.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mlib_GraphicsFillPolygon_8(3MLIB), mlib_GraphicsFillPolygon_A_8(3MLIB), mlib_GraphicsFillPolygon_AG_8(3MLIB), mlib_GraphicsFillPolygon_AGZ_8(3MLIB), mlib_GraphicsFillPolygon_AZ_8(3MLIB), mlib_GraphicsFillPolygon_G_8(3MLIB), mlib_GraphicsFillPolygon_GZ_8(3MLIB), mlib_GraphicsFillPolygon_Z_8(3MLIB), attributes(5)

mllib_GraphicsFillPolygon_Z_8(3MLIB)

NAME	mllib_GraphicsFillPolygon_Z_8, mllib_GraphicsFillPolygon_Z_32 – draw filled polygon with Z buffering														
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsFillPolygon_Z_8(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, const mllib_s16 *<i>z</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>); mllib_status mllib_GraphicsFillPolygon_Z_32(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, const mllib_s16 *<i>z</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>);</pre>														
DESCRIPTION	Each of these functions draws a filled polygon enclosing (x1,y1), (x2,y2), ..., and (xn,yn).														
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>zbuffer</i></td><td>Pointer to the image that holds the Z buffer.</td></tr><tr><td><i>x</i></td><td>Pointer to the array of X coordinates of the vertices.</td></tr><tr><td><i>y</i></td><td>Pointer to the array of Y coordinates of the vertices.</td></tr><tr><td><i>z</i></td><td>Pointer to the array of Z coordinates of the vertices.</td></tr><tr><td><i>npoints</i></td><td>Number of vertices in the arrays.</td></tr><tr><td><i>c</i></td><td>Color used in the drawing.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>zbuffer</i>	Pointer to the image that holds the Z buffer.	<i>x</i>	Pointer to the array of X coordinates of the vertices.	<i>y</i>	Pointer to the array of Y coordinates of the vertices.	<i>z</i>	Pointer to the array of Z coordinates of the vertices.	<i>npoints</i>	Number of vertices in the arrays.	<i>c</i>	Color used in the drawing.
<i>buffer</i>	Pointer to the image into which the function is drawing.														
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.														
<i>x</i>	Pointer to the array of X coordinates of the vertices.														
<i>y</i>	Pointer to the array of Y coordinates of the vertices.														
<i>z</i>	Pointer to the array of Z coordinates of the vertices.														
<i>npoints</i>	Number of vertices in the arrays.														
<i>c</i>	Color used in the drawing.														
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.														
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe								
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														
MT-Level	MT-Safe														
SEE ALSO	mllib_GraphicsFillPolygon_8(3MLIB), mllib_GraphicsFillPolygon_A_8(3MLIB), mllib_GraphicsFillPolygon_AG_8(3MLIB), mllib_GraphicsFillPolygon_AGZ_8(3MLIB), mllib_GraphicsFillPolygon_AZ_8(3MLIB), mllib_GraphicsFillPolygon_G_8(3MLIB), mllib_GraphicsFillPolygon_GZ_8(3MLIB), mllib_GraphicsFillPolygon_X_8(3MLIB), attributes(5)														

mllib_GraphicsFillRectangle_8(3MLIB)

NAME mllib_GraphicsFillRectangle_8, mllib_GraphicsFillRectangle_32 – draw filled rectangle

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
mllib_status mllib_GraphicsFillRectangle_8(mllib_image *buffer,  
      mllib_s16 x, mllib_s16 y, mllib_s32 w, mllib_s32 h, mllib_s32 c);  
  
mllib_status mllib_GraphicsFillRectangle_32(mllib_image *buffer,  
      mllib_s16 x, mllib_s16 y, mllib_s32 w, mllib_s32 h, mllib_s32 c);
```

DESCRIPTION Each of these functions draws a filled rectangle with the upper-left corner at (x,y), width w, and height h.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.
x X coordinate of the upper-left corner of the rectangle.
y Y coordinate of the upper-left corner of the rectangle.
w Width of the rectangle.
h Height of the rectangle.
c Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_GraphicsFillRectangle_X_8\(3MLIB\)](#), [attributes\(5\)](#)

mllib_GraphicsFillRectangle_X_8(3MLIB)

NAME	mllib_GraphicsFillRectangle_X_8, mllib_GraphicsFillRectangle_X_32 – draw filled rectangle in Xor mode														
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsFillRectangle_X_8(mllib_image *<i>buffer</i>, mllib_s16 <i>x</i>, mllib_s16 <i>y</i>, mllib_s32 <i>w</i>, mllib_s32 <i>h</i>, mllib_s32 <i>c</i>, mllib_s32 <i>c2</i>); mllib_status mllib_GraphicsFillRectangle_X_32(mllib_image *<i>buffer</i>, mllib_s16 <i>x</i>, mllib_s16 <i>y</i>, mllib_s32 <i>w</i>, mllib_s32 <i>h</i>, mllib_s32 <i>c</i>, mllib_s32 <i>c2</i>);</pre>														
DESCRIPTION	Each of these functions draws a filled rectangle with the upper-left corner at (x,y), width w, and height h.														
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>x</i></td><td>X coordinate of the upper-left corner of the rectangle.</td></tr><tr><td><i>y</i></td><td>Y coordinate of the upper-left corner of the rectangle.</td></tr><tr><td><i>w</i></td><td>Width of the rectangle.</td></tr><tr><td><i>h</i></td><td>Height of the rectangle.</td></tr><tr><td><i>c</i></td><td>Color used in the drawing.</td></tr><tr><td><i>c2</i></td><td>Alternation color.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>x</i>	X coordinate of the upper-left corner of the rectangle.	<i>y</i>	Y coordinate of the upper-left corner of the rectangle.	<i>w</i>	Width of the rectangle.	<i>h</i>	Height of the rectangle.	<i>c</i>	Color used in the drawing.	<i>c2</i>	Alternation color.
<i>buffer</i>	Pointer to the image into which the function is drawing.														
<i>x</i>	X coordinate of the upper-left corner of the rectangle.														
<i>y</i>	Y coordinate of the upper-left corner of the rectangle.														
<i>w</i>	Width of the rectangle.														
<i>h</i>	Height of the rectangle.														
<i>c</i>	Color used in the drawing.														
<i>c2</i>	Alternation color.														
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.														
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe								
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														
MT-Level	MT-Safe														
SEE ALSO	mllib_GraphicsFillRectangle_8(3MLIB) , <code>attributes(5)</code>														

mllib_GraphicsFillTriangle_8(3MLIB)

NAME mllib_GraphicsFillTriangle_8, mllib_GraphicsFillTriangle_32 – draw filled triangle

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillTriangle_8(mllib_image *buffer,
      mllib_s16 x1, mllib_s16 y1, mllib_s16 x2, mllib_s16 y2, mllib_s16
      x3, mllib_s16 y3, mllib_s32 c);

mllib_status mllib_GraphicsFillTriangle_32(mllib_image *buffer,
      mllib_s16 x1, mllib_s16 y1, mllib_s16 x2, mllib_s16 y2, mllib_s16
      x3, mllib_s16 y3, mllib_s32 c);
```

DESCRIPTION Each of these functions draws a filled triangle with the vertices at (x1,y1), (x2,y2), and (x3,y3).

PARAMETERS Each of the functions takes the following arguments:

<i>buffer</i>	Pointer to the image into which the function is drawing.
<i>x1</i>	X coordinate of the first vertex.
<i>y1</i>	Y coordinate of the first vertex.
<i>x2</i>	X coordinate of the second vertex.
<i>y2</i>	Y coordinate of the second vertex.
<i>x3</i>	X coordinate of the third vertex.
<i>y3</i>	Y coordinate of the third vertex.
<i>c</i>	Color used in the drawing.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsFillTriangle_A_8(3MLIB), mllib_GraphicsFillTriangle_AG_8(3MLIB), mllib_GraphicsFillTriangle_AGZ_8(3MLIB), mllib_GraphicsFillTriangle_AZ_8(3MLIB), mllib_GraphicsFillTriangle_G_8(3MLIB), mllib_GraphicsFillTriangle_GZ_8(3MLIB), mllib_GraphicsFillTriangle_X_8(3MLIB), mllib_GraphicsFillTriangle_Z_8(3MLIB), attributes(5)

mllib_GraphicsFillTriangle_A_8(3MLIB)

NAME	mllib_GraphicsFillTriangle_A_8, mllib_GraphicsFillTriangle_A_32 – draw filled triangle with antialiasing																
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsFillTriangle_A_8(mllib_image *<i>buffer</i>, mllib_s16 <i>x1</i>, mllib_s16 <i>y1</i>, mllib_s16 <i>x2</i>, mllib_s16 <i>y2</i>, mllib_s16 <i>x3</i>, mllib_s16 <i>y3</i>, mllib_s32 <i>c</i>); mllib_status mllib_GraphicsFillTriangle_A_32(mllib_image *<i>buffer</i>, mllib_s16 <i>x1</i>, mllib_s16 <i>y1</i>, mllib_s16 <i>x2</i>, mllib_s16 <i>y2</i>, mllib_s16 <i>x3</i>, mllib_s16 <i>y3</i>, mllib_s32 <i>c</i>);</pre>																
DESCRIPTION	Each of these functions draws a filled triangle with the vertices at (x1,y1), (x2,y2), and (x3,y3).																
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <table border="0"> <tr> <td style="padding-right: 20px;"><i>buffer</i></td> <td>Pointer to the image into which the function is drawing.</td> </tr> <tr> <td><i>x1</i></td> <td>X coordinate of the first vertex.</td> </tr> <tr> <td><i>y1</i></td> <td>Y coordinate of the first vertex.</td> </tr> <tr> <td><i>x2</i></td> <td>X coordinate of the second vertex.</td> </tr> <tr> <td><i>y2</i></td> <td>Y coordinate of the second vertex.</td> </tr> <tr> <td><i>x3</i></td> <td>X coordinate of the third vertex.</td> </tr> <tr> <td><i>y3</i></td> <td>Y coordinate of the third vertex.</td> </tr> <tr> <td><i>c</i></td> <td>Color used in the drawing.</td> </tr> </table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>x1</i>	X coordinate of the first vertex.	<i>y1</i>	Y coordinate of the first vertex.	<i>x2</i>	X coordinate of the second vertex.	<i>y2</i>	Y coordinate of the second vertex.	<i>x3</i>	X coordinate of the third vertex.	<i>y3</i>	Y coordinate of the third vertex.	<i>c</i>	Color used in the drawing.
<i>buffer</i>	Pointer to the image into which the function is drawing.																
<i>x1</i>	X coordinate of the first vertex.																
<i>y1</i>	Y coordinate of the first vertex.																
<i>x2</i>	X coordinate of the second vertex.																
<i>y2</i>	Y coordinate of the second vertex.																
<i>x3</i>	X coordinate of the third vertex.																
<i>y3</i>	Y coordinate of the third vertex.																
<i>c</i>	Color used in the drawing.																
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.																
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:																
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe										
ATTRIBUTE TYPE	ATTRIBUTE VALUE																
Interface Stability	Evolving																
MT-Level	MT-Safe																
SEE ALSO	<p>mllib_GraphicsFillTriangle_8(3MLIB), mllib_GraphicsFillTriangle_AG_8(3MLIB), mllib_GraphicsFillTriangle_AGZ_8(3MLIB), mllib_GraphicsFillTriangle_AZ_8(3MLIB), mllib_GraphicsFillTriangle_G_8(3MLIB), mllib_GraphicsFillTriangle_GZ_8(3MLIB), mllib_GraphicsFillTriangle_X_8(3MLIB), mllib_GraphicsFillTriangle_Z_8(3MLIB), attributes(5)</p>																

mllib_GraphicsFillTriangle_AG_8(3MLIB)

NAME mllib_GraphicsFillTriangle_AG_8, mllib_GraphicsFillTriangle_AG_32 – draw line with antialiasing and Gouraud shading

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillTriangle_AG_8(mllib_image *buffer,
      mllib_s16 x1, mllib_s16 y1, mllib_s16 x2, mllib_s16 y2, mllib_s16
      x3, mllib_s16 y3, mllib_s32 c1, mllib_s32 c2, mllib_s32 c3);

mllib_status mllib_GraphicsFillTriangle_AG_32(mllib_image *buffer,
      mllib_s16 x1, mllib_s16 y1, mllib_s16 x2, mllib_s16 y2, mllib_s16
      x3, mllib_s16 y3, mllib_s32 c1, mllib_s32 c2, mllib_s32 c3);
```

DESCRIPTION Each of these functions draws a filled triangle with the vertices at (x1,y1), (x2,y2), and (x3,y3).

PARAMETERS Each of the functions takes the following arguments:

<i>buffer</i>	Pointer to the image into which the function is drawing.
<i>x1</i>	X coordinate of the first vertex.
<i>y1</i>	Y coordinate of the first vertex.
<i>x2</i>	X coordinate of the second vertex.
<i>y2</i>	Y coordinate of the second vertex.
<i>x3</i>	X coordinate of the third vertex.
<i>y3</i>	Y coordinate of the third vertex.
<i>c1</i>	Color of the first vertex.
<i>c2</i>	Color of the second vertex.
<i>c3</i>	Color of the third vertex.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsFillTriangle_8(3MLIB), mllib_GraphicsFillTriangle_A_8(3MLIB), mllib_GraphicsFillTriangle_AGZ_8(3MLIB), mllib_GraphicsFillTriangle_AZ_8(3MLIB), mllib_GraphicsFillTriangle_G_8(3MLIB),

`mllib_GraphicsFillTriangle_AG_8(3MLIB)`

```
mllib_GraphicsFillTriangle_GZ_8(3MLIB),  
mllib_GraphicsFillTriangle_X_8(3MLIB),  
mllib_GraphicsFillTriangle_Z_8(3MLIB), attributes(5)
```

mllib_GraphicsFillTriangle_AGZ_8(3MLIB)

NAME	mllib_GraphicsFillTriangle_AGZ_8, mllib_GraphicsFillTriangle_AGZ_32 – draw filled triangle with antialiasing, Gouraud shading, and Z buffering																												
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_GraphicsFillTriangle_AGZ_8(mllib_image *buffer, mllib_image *zbuffer, mllib_s16 x1, mllib_s16 y1, mllib_s16 z1, mllib_s16 x2, mllib_s16 y2, mllib_s16 z2, mllib_s16 x3, mllib_s16 y3, mllib_s16 z3, mllib_s32 c1, mllib_s32 c2, mllib_s32 c3); mllib_status mllib_GraphicsFillTriangle_AGZ_32(mllib_image *buffer, mllib_image *zbuffer, mllib_s16 x1, mllib_s16 y1, mllib_s16 z1, mllib_s16 x2, mllib_s16 y2, mllib_s16 z2, mllib_s16 x3, mllib_s16 y3, mllib_s16 z3, mllib_s32 c1, mllib_s32 c2, mllib_s32 c3);</pre>																												
DESCRIPTION	Each of these functions draws a triangle with the vertices at (x1,y1), (x2,y2), and (x3,y3).																												
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <table border="0"> <tr> <td style="padding-right: 20px;"><i>buffer</i></td> <td>Pointer to the image into which the function is drawing.</td> </tr> <tr> <td><i>zbuffer</i></td> <td>Pointer to the image that holds the Z buffer.</td> </tr> <tr> <td><i>x1</i></td> <td>X coordinate of the first vertex.</td> </tr> <tr> <td><i>y1</i></td> <td>Y coordinate of the first vertex.</td> </tr> <tr> <td><i>z1</i></td> <td>Z coordinate of the first vertex.</td> </tr> <tr> <td><i>x2</i></td> <td>X coordinate of the second vertex.</td> </tr> <tr> <td><i>y2</i></td> <td>Y coordinate of the second vertex.</td> </tr> <tr> <td><i>z2</i></td> <td>Z coordinate of the second vertex.</td> </tr> <tr> <td><i>x3</i></td> <td>X coordinate of the third vertex.</td> </tr> <tr> <td><i>y3</i></td> <td>Y coordinate of the third vertex.</td> </tr> <tr> <td><i>z3</i></td> <td>Z coordinate of the third vertex.</td> </tr> <tr> <td><i>c1</i></td> <td>Color of the first vertex.</td> </tr> <tr> <td><i>c2</i></td> <td>Color of the second vertex.</td> </tr> <tr> <td><i>c3</i></td> <td>Color of the third vertex.</td> </tr> </table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>zbuffer</i>	Pointer to the image that holds the Z buffer.	<i>x1</i>	X coordinate of the first vertex.	<i>y1</i>	Y coordinate of the first vertex.	<i>z1</i>	Z coordinate of the first vertex.	<i>x2</i>	X coordinate of the second vertex.	<i>y2</i>	Y coordinate of the second vertex.	<i>z2</i>	Z coordinate of the second vertex.	<i>x3</i>	X coordinate of the third vertex.	<i>y3</i>	Y coordinate of the third vertex.	<i>z3</i>	Z coordinate of the third vertex.	<i>c1</i>	Color of the first vertex.	<i>c2</i>	Color of the second vertex.	<i>c3</i>	Color of the third vertex.
<i>buffer</i>	Pointer to the image into which the function is drawing.																												
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.																												
<i>x1</i>	X coordinate of the first vertex.																												
<i>y1</i>	Y coordinate of the first vertex.																												
<i>z1</i>	Z coordinate of the first vertex.																												
<i>x2</i>	X coordinate of the second vertex.																												
<i>y2</i>	Y coordinate of the second vertex.																												
<i>z2</i>	Z coordinate of the second vertex.																												
<i>x3</i>	X coordinate of the third vertex.																												
<i>y3</i>	Y coordinate of the third vertex.																												
<i>z3</i>	Z coordinate of the third vertex.																												
<i>c1</i>	Color of the first vertex.																												
<i>c2</i>	Color of the second vertex.																												
<i>c3</i>	Color of the third vertex.																												
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.																												
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:																												

mllib_GraphicsFillTriangle_AGZ_8(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

mllib_GraphicsFillTriangle_8(3MLIB),
mllib_GraphicsFillTriangle_A_8(3MLIB),
mllib_GraphicsFillTriangle_AG_8(3MLIB),
mllib_GraphicsFillTriangle_AZ_8(3MLIB),
mllib_GraphicsFillTriangle_G_8(3MLIB),
mllib_GraphicsFillTriangle_GZ_8(3MLIB),
mllib_GraphicsFillTriangle_X_8(3MLIB),
mllib_GraphicsFillTriangle_Z_8(3MLIB), attributes(5)

NAME mllib_GraphicsFillTriangle_AZ_8, mllib_GraphicsFillTriangle_AZ_32 – draw filled triangle with antialiasing and Z buffering

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillTriangle_AZ_8(mllib_image *buffer,
      mllib_image *zbuffer, mllib_s16 x1, mllib_s16 y1, mllib_s16 z1,
      mllib_s16 x2, mllib_s16 y2, mllib_s16 z2, mllib_s16 x3, mllib_s16
      y3, mllib_s16 z3, mllib_s32 c);

mllib_status mllib_GraphicsFillTriangle_AZ_32(mllib_image *buffer,
      mllib_image *zbuffer, mllib_s16 x1, mllib_s16 y1, mllib_s16 z1,
      mllib_s16 x2, mllib_s16 y2, mllib_s16 z2, mllib_s16 x3, mllib_s16
      y3, mllib_s16 z3, mllib_s32 c);
```

DESCRIPTION Each of these functions draws a filled triangle with the vertices at (x1,y1), (x2,y2), and (x3,y3).

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

zbuffer Pointer to the image that holds the Z buffer.

x1 X coordinate of the first vertex.

y1 Y coordinate of the first vertex.

z1 Z coordinate of the first vertex.

x2 X coordinate of the second vertex.

y2 Y coordinate of the second vertex.

z2 Z coordinate of the second vertex.

x3 X coordinate of the third vertex.

y3 Y coordinate of the third vertex.

z3 Z coordinate of the third vertex.

c Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

mllib_GraphicsFillTriangle_AZ_8(3MLIB)

SEE ALSO | mllib_GraphicsFillTriangle_8(3MLIB),
mllib_GraphicsFillTriangle_A_8(3MLIB),
mllib_GraphicsFillTriangle_AG_8(3MLIB),
mllib_GraphicsFillTriangle_AGZ_8(3MLIB),
mllib_GraphicsFillTriangle_G_8(3MLIB),
mllib_GraphicsFillTriangle_GZ_8(3MLIB),
mllib_GraphicsFillTriangle_X_8(3MLIB),
mllib_GraphicsFillTriangle_Z_8(3MLIB), attributes(5)

mllib_GraphicsFillTriangleFanSet_8(3MLIB)

NAME mllib_GraphicsFillTriangleFanSet_8, mllib_GraphicsFillTriangleFanSet_32 – draw filled triangle set where all members of the set have a common vertex

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillTriangleFanSet_8(mllib_image *buffer,
        const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
        mllib_s32 c);

mllib_status mllib_GraphicsFillTriangleFanSet_32(mllib_image *buffer,
        const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
        mllib_s32 c);
```

DESCRIPTION Each of these functions draws a filled set of triangles with vertices at {(x1,y1), (x2,y2),(x3,y3)}, {(x1,y1),(x3,y3),(x4,y4)}, ..., and {(x1,y1),(xn-1,yn-1), (xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.
x Pointer to array of X coordinates of the points.
y Pointer to array of Y coordinates of the points.
npoints Number of points in the arrays.
c Color used in the drawing.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsFillTriangleFanSet_A_8(3MLIB),
mllib_GraphicsFillTriangleFanSet_AG_8(3MLIB),
mllib_GraphicsFillTriangleFanSet_AGZ_8(3MLIB),
mllib_GraphicsFillTriangleFanSet_AZ_8(3MLIB),
mllib_GraphicsFillTriangleFanSet_G_8(3MLIB),
mllib_GraphicsFillTriangleFanSet_GZ_8(3MLIB),
mllib_GraphicsFillTriangleFanSet_X_8(3MLIB),
mllib_GraphicsFillTriangleFanSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsFillTriangleFanSet_A_8(3MLIB)

NAME	mllib_GraphicsFillTriangleFanSet_A_8, mllib_GraphicsFillTriangleFanSet_A_32 – draw filled triangle set with antialiasing where all members of the set have a common vertex										
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsFillTriangleFanSet_A_8(mllib_image *buffer, const mlib_s16 *x, const mlib_s16 *y, mlib_s32 npoints, mlib_s32 c); mllib_status mllib_GraphicsFillTriangleFanSet_A_32(mllib_image *buffer, const mlib_s16 *x, const mlib_s16 *y, mlib_s32 npoints, mlib_s32 c);</pre>										
DESCRIPTION	Each of these functions draws a filled set of triangles with vertices at {(x1,y1), (x2,y2),(x3,y3)}, {(x1,y1),(x3,y3),(x4,y4)}, ..., and {(x1,y1),(xn-1,yn-1), (xn,yn)}.										
PARAMETERS	Each of the functions takes the following arguments: <table border="0" style="margin-left: 20px;"> <tr> <td style="padding-right: 20px;"><i>buffer</i></td> <td>Pointer to the image into which the function is drawing.</td> </tr> <tr> <td><i>x</i></td> <td>Pointer to array of X coordinates of the points.</td> </tr> <tr> <td><i>y</i></td> <td>Pointer to array of Y coordinates of the points.</td> </tr> <tr> <td><i>npoints</i></td> <td>Number of points in the arrays.</td> </tr> <tr> <td><i>c</i></td> <td>Color used in the drawing.</td> </tr> </table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>x</i>	Pointer to array of X coordinates of the points.	<i>y</i>	Pointer to array of Y coordinates of the points.	<i>npoints</i>	Number of points in the arrays.	<i>c</i>	Color used in the drawing.
<i>buffer</i>	Pointer to the image into which the function is drawing.										
<i>x</i>	Pointer to array of X coordinates of the points.										
<i>y</i>	Pointer to array of Y coordinates of the points.										
<i>npoints</i>	Number of points in the arrays.										
<i>c</i>	Color used in the drawing.										
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.										
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1" style="margin-left: 20px; width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe				
ATTRIBUTE TYPE	ATTRIBUTE VALUE										
Interface Stability	Evolving										
MT-Level	MT-Safe										
SEE ALSO	mllib_GraphicsFillTriangleFanSet_8(3MLIB), mllib_GraphicsFillTriangleFanSet_AG_8(3MLIB), mllib_GraphicsFillTriangleFanSet_AGZ_8(3MLIB), mllib_GraphicsFillTriangleFanSet_AZ_8(3MLIB), mllib_GraphicsFillTriangleFanSet_G_8(3MLIB), mllib_GraphicsFillTriangleFanSet_GZ_8(3MLIB), mllib_GraphicsFillTriangleFanSet_X_8(3MLIB), mllib_GraphicsFillTriangleFanSet_Z_8(3MLIB), attributes(5)										

mllib_GraphicsFillTriangleFanSet_AG_8(3MLIB)

NAME mllib_GraphicsFillTriangleFanSet_AG_8, mllib_GraphicsFillTriangleFanSet_AG_32 – draw filled triangle set with antialiasing and gouraud shading, where all members of the set have a common vertex

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillTriangleFanSet_AG_8(mllib_image
    *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
    mllib_s32 *c);

mllib_status mllib_GraphicsFillTriangleFanSet_AG_32(mllib_image
    *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
    mllib_s32 *c);
```

DESCRIPTION Each of these functions draws a filled set of triangles with vertices at {(x1,y1), (x2,y2),(x3,y3)}, {(x1,y1),(x3,y3),(x4,y4)}, ..., and {(x1,y1),(xn-1,yn-1), (xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.
x Pointer to array of X coordinates of the points.
y Pointer to array of Y coordinates of the points.
npoints Number of points in the arrays.
c Pointer to array of colors of the points.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsFillTriangleFanSet_8(3MLIB), mllib_GraphicsFillTriangleFanSet_A_8(3MLIB), mllib_GraphicsFillTriangleFanSet_AGZ_8(3MLIB), mllib_GraphicsFillTriangleFanSet_AZ_8(3MLIB), mllib_GraphicsFillTriangleFanSet_G_8(3MLIB), mllib_GraphicsFillTriangleFanSet_GZ_8(3MLIB), mllib_GraphicsFillTriangleFanSet_X_8(3MLIB), mllib_GraphicsFillTriangleFanSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsFillTriangleFanSet_AGZ_8(3MLIB)

NAME	mllib_GraphicsFillTriangleFanSet_AGZ_8, mllib_GraphicsFillTriangleFanSet_AGZ_32 – draw filled triangle set with antialiasing, Gouraud shading, and Z buffering, where all members of the set have a common vertex														
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_GraphicsFillTriangleFanSet_AGZ_8(mllib_image *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c); mllib_status mllib_GraphicsFillTriangleFanSet_AGZ_32(mllib_image *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);</pre>														
DESCRIPTION	Each of these functions draws a filled set of triangles with vertices at {(x1,y1), (x2,y2),(x3,y3)}, {(x1,y1),(x3,y3),(x4,y4)}, ..., and {(x1,y1),(xn-1,yn-1), (xn,yn)}.														
PARAMETERS	Each of the functions takes the following arguments: <table border="0" style="margin-left: 20px;"> <tr> <td style="padding-right: 20px;"><i>buffer</i></td> <td>Pointer to the image into which the function is drawing.</td> </tr> <tr> <td style="padding-right: 20px;"><i>zbuffer</i></td> <td>Pointer to the image that holds the Z buffer.</td> </tr> <tr> <td style="padding-right: 20px;"><i>x</i></td> <td>Pointer to array of X coordinates of the points.</td> </tr> <tr> <td style="padding-right: 20px;"><i>y</i></td> <td>Pointer to array of Y coordinates of the points.</td> </tr> <tr> <td style="padding-right: 20px;"><i>z</i></td> <td>Pointer to array of Z coordinates of the points.</td> </tr> <tr> <td style="padding-right: 20px;"><i>npoints</i></td> <td>Number of points in the arrays.</td> </tr> <tr> <td style="padding-right: 20px;"><i>c</i></td> <td>Pointer to array of colors of the points.</td> </tr> </table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>zbuffer</i>	Pointer to the image that holds the Z buffer.	<i>x</i>	Pointer to array of X coordinates of the points.	<i>y</i>	Pointer to array of Y coordinates of the points.	<i>z</i>	Pointer to array of Z coordinates of the points.	<i>npoints</i>	Number of points in the arrays.	<i>c</i>	Pointer to array of colors of the points.
<i>buffer</i>	Pointer to the image into which the function is drawing.														
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.														
<i>x</i>	Pointer to array of X coordinates of the points.														
<i>y</i>	Pointer to array of Y coordinates of the points.														
<i>z</i>	Pointer to array of Z coordinates of the points.														
<i>npoints</i>	Number of points in the arrays.														
<i>c</i>	Pointer to array of colors of the points.														
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.														
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1" style="margin-left: 20px; width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe								
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														
MT-Level	MT-Safe														
SEE ALSO	mllib_GraphicsFillTriangleFanSet_8(3MLIB), mllib_GraphicsFillTriangleFanSet_A_8(3MLIB), mllib_GraphicsFillTriangleFanSet_AG_8(3MLIB), mllib_GraphicsFillTriangleFanSet_AZ_8(3MLIB), mllib_GraphicsFillTriangleFanSet_G_8(3MLIB), mllib_GraphicsFillTriangleFanSet_GZ_8(3MLIB), mllib_GraphicsFillTriangleFanSet_X_8(3MLIB), mllib_GraphicsFillTriangleFanSet_Z_8(3MLIB), attributes(5)														

mllib_GraphicsFillTriangleFanSet_AZ_8(3MLIB)

NAME mllib_GraphicsFillTriangleFanSet_AZ_8, mllib_GraphicsFillTriangleFanSet_AZ_32 – draw filled triangle set with antialiasing and Z buffering, where all members of the set have a common vertex

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillTriangleFanSet_AZ_8(mllib_image
    *buffer, mllib_image *zbuffer, const mlib_s16 *x, const mlib_s16
    *y, const mlib_s16 *z, mlib_s32 npoints, mlib_s32 c);

mllib_status mllib_GraphicsFillTriangleFanSet_AZ_32(mllib_image
    *buffer, mllib_image *zbuffer, const mlib_s16 *x, const mlib_s16
    *y, const mlib_s16 *z, mlib_s32 npoints, mlib_s32 c);
```

DESCRIPTION Each of these functions draws a filled set of triangles with vertices at {(x1,y1), (x2,y2),(x3,y3)}, {(x1,y1),(x3,y3),(x4,y4)}, ..., and {(x1,y1),(xn-1,yn-1), (xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

<i>buffer</i>	Pointer to the image into which the function is drawing.
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.
<i>x</i>	Pointer to array of X coordinates of the points.
<i>y</i>	Pointer to array of Y coordinates of the points.
<i>z</i>	Pointer to array of Z coordinates of the points.
<i>npoints</i>	Number of points in the arrays.
<i>c</i>	Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsFillTriangleFanSet_8(3MLIB), mllib_GraphicsFillTriangleFanSet_A_8(3MLIB), mllib_GraphicsFillTriangleFanSet_AG_8(3MLIB), mllib_GraphicsFillTriangleFanSet_AGZ_8(3MLIB), mllib_GraphicsFillTriangleFanSet_G_8(3MLIB), mllib_GraphicsFillTriangleFanSet_GZ_8(3MLIB), mllib_GraphicsFillTriangleFanSet_X_8(3MLIB), mllib_GraphicsFillTriangleFanSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsFillTriangleFanSet_G_8(3MLIB)

NAME	mllib_GraphicsFillTriangleFanSet_G_8, mllib_GraphicsFillTriangleFanSet_G_32 – draw filled triangle set with Gouraud shading where all members of the set have a common vertex										
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_GraphicsFillTriangleFanSet_G_8(mllib_image *buffer, const mlib_s16 *x, const mlib_s16 *y, mlib_s32 npoints, const mlib_s32 *c); mllib_status mllib_GraphicsFillTriangleFanSet_G_32(mllib_image *buffer, const mlib_s16 *x, const mlib_s16 *y, mlib_s32 npoints, const mlib_s32 *c);</pre>										
DESCRIPTION	Each of these functions draws a filled set of triangles with vertices at {(x1,y1), (x2,y2),(x3,y3)}, {(x1,y1),(x3,y3),(x4,y4)}, ..., and {(x1,y1),(xn-1,yn-1), (xn,yn)}.										
PARAMETERS	Each of the functions takes the following arguments: <table border="0" style="margin-left: 20px;"> <tr> <td style="padding-right: 20px;"><i>buffer</i></td> <td>Pointer to the image into which the function is drawing.</td> </tr> <tr> <td><i>x</i></td> <td>Pointer to array of X coordinates of the points.</td> </tr> <tr> <td><i>y</i></td> <td>Pointer to array of Y coordinates of the points.</td> </tr> <tr> <td><i>npoints</i></td> <td>Number of points in the arrays.</td> </tr> <tr> <td><i>c</i></td> <td>Pointer to array of colors of the points.</td> </tr> </table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>x</i>	Pointer to array of X coordinates of the points.	<i>y</i>	Pointer to array of Y coordinates of the points.	<i>npoints</i>	Number of points in the arrays.	<i>c</i>	Pointer to array of colors of the points.
<i>buffer</i>	Pointer to the image into which the function is drawing.										
<i>x</i>	Pointer to array of X coordinates of the points.										
<i>y</i>	Pointer to array of Y coordinates of the points.										
<i>npoints</i>	Number of points in the arrays.										
<i>c</i>	Pointer to array of colors of the points.										
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.										
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1" style="margin-left: 20px; width: 100%;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe				
ATTRIBUTE TYPE	ATTRIBUTE VALUE										
Interface Stability	Evolving										
MT-Level	MT-Safe										
SEE ALSO	mllib_GraphicsFillTriangleFanSet_8(3MLIB), mllib_GraphicsFillTriangleFanSet_A_8(3MLIB), mllib_GraphicsFillTriangleFanSet_AG_8(3MLIB), mllib_GraphicsFillTriangleFanSet_AGZ_8(3MLIB), mllib_GraphicsFillTriangleFanSet_AZ_8(3MLIB), mllib_GraphicsFillTriangleFanSet_GZ_8(3MLIB), mllib_GraphicsFillTriangleFanSet_X_8(3MLIB), mllib_GraphicsFillTriangleFanSet_Z_8(3MLIB), attributes(5)										

mllib_GraphicsFillTriangleFanSet_GZ_8(3MLIB)

NAME mllib_GraphicsFillTriangleFanSet_GZ_8, mllib_GraphicsFillTriangleFanSet_GZ_32 – draw filled triangle set with Gouraud shading and Z buffering, where all members of the set have a common vertex

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillTriangleFanSet_GZ_8(mllib_image
    *buffer, mllib_image *zbuffer, const mlib_s16 *x, const mlib_s16
    *y, const mlib_s16 *z, mlib_s32 npoints, const mlib_s32 *c);

mllib_status mllib_GraphicsFillTriangleFanSet_GZ_32(mllib_image
    *buffer, mllib_image *zbuffer, const mlib_s16 *x, const mlib_s16
    *y, const mlib_s16 *z, mlib_s32 npoints, const mlib_s32 *c);
```

DESCRIPTION Each of these functions draws a filled set of triangles with vertices at {(x1,y1), (x2,y2),(x3,y3)}, {(x1,y1),(x3,y3),(x4,y4)}, ..., and {(x1,y1),(xn-1,yn-1), (xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

<i>buffer</i>	Pointer to the image into which the function is drawing.
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.
<i>x</i>	Pointer to array of X coordinates of the points.
<i>y</i>	Pointer to array of Y coordinates of the points.
<i>z</i>	Pointer to array of Z coordinates of the points.
<i>npoints</i>	Number of points in the arrays.
<i>c</i>	Pointer to array of colors of the points.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsFillTriangleFanSet_8(3MLIB), mllib_GraphicsFillTriangleFanSet_A_8(3MLIB), mllib_GraphicsFillTriangleFanSet_AG_8(3MLIB), mllib_GraphicsFillTriangleFanSet_AGZ_8(3MLIB), mllib_GraphicsFillTriangleFanSet_AZ_8(3MLIB), mllib_GraphicsFillTriangleFanSet_G_8(3MLIB), mllib_GraphicsFillTriangleFanSet_X_8(3MLIB), mllib_GraphicsFillTriangleFanSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsFillTriangleFanSet_X_8(3MLIB)

NAME	mllib_GraphicsFillTriangleFanSet_X_8, mllib_GraphicsFillTriangleFanSet_X_32 – draw filled triangle set in Xor mode where all members of the set have a common vertex						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsFillTriangleFanSet_X_8(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c, mllib_s32 c2); mllib_status mllib_GraphicsFillTriangleFanSet_X_32(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c, mllib_s32 c2);</pre>						
DESCRIPTION	Each of these functions draws a filled set of triangles with vertices at {(x1,y1), (x2,y2),(x3,y3)}, {(x1,y1),(x3,y3),(x4,y4)}, ..., and {(x1,y1),(xn-1,yn-1), (xn,yn)}.						
PARAMETERS	Each of the functions takes the following arguments: <i>buffer</i> Pointer to the image into which the function is drawing. <i>x</i> Pointer to array of X coordinates of the points. <i>y</i> Pointer to array of Y coordinates of the points. <i>npoints</i> Number of points in the arrays. <i>c</i> Color used in the drawing. <i>c2</i> Alternation color.						
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_GraphicsFillTriangleFanSet_8(3MLIB), mllib_GraphicsFillTriangleFanSet_A_8(3MLIB), mllib_GraphicsFillTriangleFanSet_AG_8(3MLIB), mllib_GraphicsFillTriangleFanSet_AGZ_8(3MLIB), mllib_GraphicsFillTriangleFanSet_AZ_8(3MLIB), mllib_GraphicsFillTriangleFanSet_G_8(3MLIB), mllib_GraphicsFillTriangleFanSet_GZ_8(3MLIB), mllib_GraphicsFillTriangleFanSet_Z_8(3MLIB), attributes(5)						

mllib_GraphicsFillTriangleFanSet_Z_8(3MLIB)

NAME mllib_GraphicsFillTriangleFanSet_Z_8, mllib_GraphicsFillTriangleFanSet_Z_32 – draw filled triangle set with Z buffering where all members of the set have a common vertex

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillTriangleFanSet_Z_8(mllib_image
    *buffer, mllib_image *zbuffer, mllib_s16 *x, mllib_s16 *y, mllib_s16
    *z, mllib_s32 npoints, mllib_s32 c);

mllib_status mllib_GraphicsFillTriangleFanSet_Z_32(mllib_image
    *buffer, mllib_image *zbuffer, mllib_s16 *x, mllib_s16 *y, mllib_s16
    *z, mllib_s32 npoints, mllib_s32 c);
```

DESCRIPTION Each of these functions draws a filled set of triangles with vertices at {(x1,y1), (x2,y2),(x3,y3)}, {(x1,y1),(x3,y3),(x4,y4)}, ..., and {(x1,y1),(xn-1,yn-1), (xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

<i>buffer</i>	Pointer to the image into which the function is drawing.
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.
<i>x</i>	Pointer to array of X coordinates of the points.
<i>y</i>	Pointer to array of Y coordinates of the points.
<i>z</i>	Pointer to array of Z coordinates of the points.
<i>npoints</i>	Number of points in the arrays.
<i>c</i>	Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsFillTriangleFanSet_8(3MLIB), mllib_GraphicsFillTriangleFanSet_A_8(3MLIB), mllib_GraphicsFillTriangleFanSet_AG_8(3MLIB), mllib_GraphicsFillTriangleFanSet_AGZ_8(3MLIB), mllib_GraphicsFillTriangleFanSet_AZ_8(3MLIB), mllib_GraphicsFillTriangleFanSet_G_8(3MLIB), mllib_GraphicsFillTriangleFanSet_GZ_8(3MLIB), mllib_GraphicsFillTriangleFanSet_X_8(3MLIB), attributes(5)

mllib_GraphicsFillTriangle_G_8(3MLIB)

NAME	mllib_GraphicsFillTriangle_G_8, mllib_GraphicsFillTriangle_G_32 – draw filled triangle with Gouraud shading																				
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsFillTriangle_G_8(mllib_image *<i>buffer</i>, mllib_s16 <i>x1</i>, mllib_s16 <i>y1</i>, mllib_s16 <i>x2</i>, mllib_s16 <i>y2</i>, mllib_s16 <i>x3</i>, mllib_s16 <i>y3</i>, mllib_s32 <i>c1</i>, mllib_s32 <i>c2</i>, mllib_s32 <i>c3</i>); mllib_status mllib_GraphicsFillTriangle_G_32(mllib_image *<i>buffer</i>, mllib_s16 <i>x1</i>, mllib_s16 <i>y1</i>, mllib_s16 <i>x2</i>, mllib_s16 <i>y2</i>, mllib_s16 <i>x3</i>, mllib_s16 <i>y3</i>, mllib_s32 <i>c1</i>, mllib_s32 <i>c2</i>, mllib_s32 <i>c3</i>);</pre>																				
DESCRIPTION	Each of these functions draws a filled triangle with the vertices at (x1,y1), (x2,y2), and (x3,y3).																				
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>x1</i></td><td>X coordinate of the first vertex.</td></tr><tr><td><i>y1</i></td><td>Y coordinate of the first vertex.</td></tr><tr><td><i>x2</i></td><td>X coordinate of the second vertex.</td></tr><tr><td><i>y2</i></td><td>Y coordinate of the second vertex.</td></tr><tr><td><i>x3</i></td><td>X coordinate of the third vertex.</td></tr><tr><td><i>y3</i></td><td>Y coordinate of the third vertex.</td></tr><tr><td><i>c1</i></td><td>Color of the first vertex.</td></tr><tr><td><i>c2</i></td><td>Color of the second vertex.</td></tr><tr><td><i>c3</i></td><td>Color of the third vertex.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>x1</i>	X coordinate of the first vertex.	<i>y1</i>	Y coordinate of the first vertex.	<i>x2</i>	X coordinate of the second vertex.	<i>y2</i>	Y coordinate of the second vertex.	<i>x3</i>	X coordinate of the third vertex.	<i>y3</i>	Y coordinate of the third vertex.	<i>c1</i>	Color of the first vertex.	<i>c2</i>	Color of the second vertex.	<i>c3</i>	Color of the third vertex.
<i>buffer</i>	Pointer to the image into which the function is drawing.																				
<i>x1</i>	X coordinate of the first vertex.																				
<i>y1</i>	Y coordinate of the first vertex.																				
<i>x2</i>	X coordinate of the second vertex.																				
<i>y2</i>	Y coordinate of the second vertex.																				
<i>x3</i>	X coordinate of the third vertex.																				
<i>y3</i>	Y coordinate of the third vertex.																				
<i>c1</i>	Color of the first vertex.																				
<i>c2</i>	Color of the second vertex.																				
<i>c3</i>	Color of the third vertex.																				
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.																				
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe														
ATTRIBUTE TYPE	ATTRIBUTE VALUE																				
Interface Stability	Evolving																				
MT-Level	MT-Safe																				
SEE ALSO	<code>mllib_GraphicsFillTriangle_8(3MLIB)</code> , <code>mllib_GraphicsFillTriangle_A_8(3MLIB)</code> , <code>mllib_GraphicsFillTriangle_AG_8(3MLIB)</code> , <code>mllib_GraphicsFillTriangle_AGZ_8(3MLIB)</code> , <code>mllib_GraphicsFillTriangle_AZ_8(3MLIB)</code> ,																				

`mllib_GraphicsFillTriangle_G_8(3MLIB)`

```
mllib_GraphicsFillTriangle_GZ_8(3MLIB),  
mllib_GraphicsFillTriangle_X_8(3MLIB),  
mllib_GraphicsFillTriangle_Z_8(3MLIB), attributes(5)
```

mllib_GraphicsFillTriangle_GZ_8(3MLIB)

NAME	mllib_GraphicsFillTriangle_GZ_8, mllib_GraphicsFillTriangle_GZ_32 – draw filled triangle with Gouraud shading and Z buffering																												
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsFillTriangle_GZ_8(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, mllib_s16 <i>x1</i>, mllib_s16 <i>y1</i>, mllib_s16 <i>z1</i>, mllib_s16 <i>x2</i>, mllib_s16 <i>y2</i>, mllib_s16 <i>z2</i>, mllib_s16 <i>x3</i>, mllib_s16 <i>y3</i>, mllib_s16 <i>z3</i>, mllib_s32 <i>c1</i>, mllib_s32 <i>c2</i>, mllib_s32 <i>c3</i>); mllib_status mllib_GraphicsFillTriangle_GZ_32(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, mllib_s16 <i>x1</i>, mllib_s16 <i>y1</i>, mllib_s16 <i>z1</i>, mllib_s16 <i>x2</i>, mllib_s16 <i>y2</i>, mllib_s16 <i>z2</i>, mllib_s16 <i>x3</i>, mllib_s16 <i>y3</i>, mllib_s16 <i>z3</i>, mllib_s32 <i>c1</i>, mllib_s32 <i>c2</i>, mllib_s32 <i>c3</i>);</pre>																												
DESCRIPTION	Each of these functions draws a triangle with the vertices at (x1,y1), (x2,y2), and (x3,y3).																												
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>zbuffer</i></td><td>Pointer to the image that holds the Z buffer.</td></tr><tr><td><i>x1</i></td><td>X coordinate of the first vertex.</td></tr><tr><td><i>y1</i></td><td>Y coordinate of the first vertex.</td></tr><tr><td><i>z1</i></td><td>Z coordinate of the first vertex.</td></tr><tr><td><i>x2</i></td><td>X coordinate of the second vertex.</td></tr><tr><td><i>y2</i></td><td>Y coordinate of the second vertex.</td></tr><tr><td><i>z2</i></td><td>Z coordinate of the second vertex.</td></tr><tr><td><i>x3</i></td><td>X coordinate of the third vertex.</td></tr><tr><td><i>y3</i></td><td>Y coordinate of the third vertex.</td></tr><tr><td><i>z3</i></td><td>Z coordinate of the third vertex.</td></tr><tr><td><i>c1</i></td><td>Color of the first vertex.</td></tr><tr><td><i>c2</i></td><td>Color of the second vertex.</td></tr><tr><td><i>c3</i></td><td>Color of the third vertex.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>zbuffer</i>	Pointer to the image that holds the Z buffer.	<i>x1</i>	X coordinate of the first vertex.	<i>y1</i>	Y coordinate of the first vertex.	<i>z1</i>	Z coordinate of the first vertex.	<i>x2</i>	X coordinate of the second vertex.	<i>y2</i>	Y coordinate of the second vertex.	<i>z2</i>	Z coordinate of the second vertex.	<i>x3</i>	X coordinate of the third vertex.	<i>y3</i>	Y coordinate of the third vertex.	<i>z3</i>	Z coordinate of the third vertex.	<i>c1</i>	Color of the first vertex.	<i>c2</i>	Color of the second vertex.	<i>c3</i>	Color of the third vertex.
<i>buffer</i>	Pointer to the image into which the function is drawing.																												
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.																												
<i>x1</i>	X coordinate of the first vertex.																												
<i>y1</i>	Y coordinate of the first vertex.																												
<i>z1</i>	Z coordinate of the first vertex.																												
<i>x2</i>	X coordinate of the second vertex.																												
<i>y2</i>	Y coordinate of the second vertex.																												
<i>z2</i>	Z coordinate of the second vertex.																												
<i>x3</i>	X coordinate of the third vertex.																												
<i>y3</i>	Y coordinate of the third vertex.																												
<i>z3</i>	Z coordinate of the third vertex.																												
<i>c1</i>	Color of the first vertex.																												
<i>c2</i>	Color of the second vertex.																												
<i>c3</i>	Color of the third vertex.																												
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.																												
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:																												

mllib_GraphicsFillTriangle_GZ_8(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

mllib_GraphicsFillTriangle_8(3MLIB),
mllib_GraphicsFillTriangle_A_8(3MLIB),
mllib_GraphicsFillTriangle_AG_8(3MLIB),
mllib_GraphicsFillTriangle_AGZ_8(3MLIB),
mllib_GraphicsFillTriangle_AZ_8(3MLIB),
mllib_GraphicsFillTriangle_G_8(3MLIB),
mllib_GraphicsFillTriangle_X_8(3MLIB),
mllib_GraphicsFillTriangle_Z_8(3MLIB), attributes(5)

mllib_GraphicsFillTriangleSet_8(3MLIB)

NAME	mllib_GraphicsFillTriangleSet_8, mllib_GraphicsFillTriangleSet_32 – draw filled triangle set where each member can have different vertices										
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsFillTriangleSet_8(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>); mllib_status mllib_GraphicsFillTriangleSet_32(mllib_image *<i>buffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>);</pre>										
DESCRIPTION	Each of these functions draws a filled set of triangles with vertices at {(x1,y1), (x2,y2),(x3,y3)}, {(x4,y4),(x5,y5),(x6,y6)}, ..., and {(xn-2,yn-2),(xn-1,yn-1), (xn,yn)}.										
PARAMETERS	Each of the functions takes the following arguments: <table border="0" style="margin-left: 20px;"> <tr> <td style="padding-right: 20px;"><i>buffer</i></td> <td>Pointer to the image into which the function is drawing.</td> </tr> <tr> <td><i>x</i></td> <td>Pointer to array of X coordinates of the points.</td> </tr> <tr> <td><i>y</i></td> <td>Pointer to array of Y coordinates of the points.</td> </tr> <tr> <td><i>npoints</i></td> <td>Number of points in the arrays. npoints must be a multiple of 3.</td> </tr> <tr> <td><i>c</i></td> <td>Color used in the drawing.</td> </tr> </table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>x</i>	Pointer to array of X coordinates of the points.	<i>y</i>	Pointer to array of Y coordinates of the points.	<i>npoints</i>	Number of points in the arrays. npoints must be a multiple of 3.	<i>c</i>	Color used in the drawing.
<i>buffer</i>	Pointer to the image into which the function is drawing.										
<i>x</i>	Pointer to array of X coordinates of the points.										
<i>y</i>	Pointer to array of Y coordinates of the points.										
<i>npoints</i>	Number of points in the arrays. npoints must be a multiple of 3.										
<i>c</i>	Color used in the drawing.										
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.										
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1" style="margin-left: 20px; width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe				
ATTRIBUTE TYPE	ATTRIBUTE VALUE										
Interface Stability	Evolving										
MT-Level	MT-Safe										
SEE ALSO	mllib_GraphicsFillTriangleSet_A_8(3MLIB), mllib_GraphicsFillTriangleSet_AG_8(3MLIB), mllib_GraphicsFillTriangleSet_AGZ_8(3MLIB), mllib_GraphicsFillTriangleSet_AZ_8(3MLIB), mllib_GraphicsFillTriangleSet_G_8(3MLIB), mllib_GraphicsFillTriangleSet_GZ_8(3MLIB), mllib_GraphicsFillTriangleSet_X_8(3MLIB), mllib_GraphicsFillTriangleSet_Z_8(3MLIB), attributes(5)										

mlib_GraphicsFillTriangleSet_A_8(3MLIB)

NAME mlib_GraphicsFillTriangleSet_A_8, mlib_GraphicsFillTriangleSet_A_32 – draw filled triangle set with antialiasing where each member can have different vertices

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_GraphicsFillTriangleSet_A_8(mlib_image *buffer,
        const mlib_s16 *x, const mlib_s16 *y, mlib_s32 npoints,
        mlib_s32 c);

mlib_status mlib_GraphicsFillTriangleSet_A_32(mlib_image *buffer,
        const mlib_s16 *x, const mlib_s16 *y, mlib_s32 npoints,
        mlib_s32 c);
```

DESCRIPTION Each of these functions draws a filled set of triangles with vertices at {(x1,y1), (x2,y2),(x3,y3)}, {(x4,y4),(x5,y5),(x6,y6)}, ..., and {(xn-2,yn-2),(xn-1,yn-1), (xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

x Pointer to array of X coordinates of the points.

y Pointer to array of Y coordinates of the points.

npoints Number of points in the arrays. npoints must be a multiple of 3.

c Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mlib_GraphicsFillTriangleSet_8(3MLIB), mlib_GraphicsFillTriangleSet_AG_8(3MLIB), mlib_GraphicsFillTriangleSet_AGZ_8(3MLIB), mlib_GraphicsFillTriangleSet_AZ_8(3MLIB), mlib_GraphicsFillTriangleSet_AZ_8(3MLIB), mlib_GraphicsFillTriangleSet_G_8(3MLIB), mlib_GraphicsFillTriangleSet_GZ_8(3MLIB), mlib_GraphicsFillTriangleSet_X_8(3MLIB), mlib_GraphicsFillTriangleSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsFillTriangleSet_AG_8(3MLIB)

NAME	mllib_GraphicsFillTriangleSet_AG_8, mllib_GraphicsFillTriangleSet_AG_32 – draw filled triangle set with antialiasing and Gouraud shading, where each member can have different vertices						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsFillTriangleSet_AG_8(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, const mllib_s32 *c); mllib_status mllib_GraphicsFillTriangleSet_AG_32(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, const mllib_s32 *c);</pre>						
DESCRIPTION	Each of these functions draws a filled set of triangles with vertices at {(x1,y1), (x2,y2),(x3,y3)}, {(x4,y4),(x5,y5),(x6,y6)}, ..., and {(xn-2,yn-2),(xn-1,yn-1), (xn,yn)}.						
PARAMETERS	Each of the functions takes the following arguments: <i>buffer</i> Pointer to the image into which the function is drawing. <i>x</i> Pointer to array of X coordinates of the points. <i>y</i> Pointer to array of Y coordinates of the points. <i>npoints</i> Number of points in the arrays. npoints must be a multiple of 3. <i>c</i> Pointer to array of colors of the points.						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_GraphicsFillTriangleSet_8(3MLIB), mllib_GraphicsFillTriangleSet_A_8(3MLIB), mllib_GraphicsFillTriangleSet_AGZ_8(3MLIB), mllib_GraphicsFillTriangleSet_AZ_8(3MLIB), mllib_GraphicsFillTriangleSet_G_8(3MLIB), mllib_GraphicsFillTriangleSet_GZ_8(3MLIB), mllib_GraphicsFillTriangleSet_X_8(3MLIB), mllib_GraphicsFillTriangleSet_Z_8(3MLIB), attributes(5)						

mllib_GraphicsFillTriangleSet_AGZ_8(3MLIB)

NAME mllib_GraphicsFillTriangleSet_AGZ_8, mllib_GraphicsFillTriangleSet_AGZ_32 – draw filled triangle set with antialiasing, Gouraud shading, and Z buffering, where each member can have different vertices

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillTriangleSet_AGZ_8(mllib_image *buffer,
        mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const
        mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);

mllib_status mllib_GraphicsFillTriangleSet_AGZ_32(mllib_image
        *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16
        *y, const mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);
```

DESCRIPTION Each of these functions draws a filled set of triangles with vertices at {(x1,y1), (x2,y2),(x3,y3)}, {(x4,y4),(x5,y5),(x6,y6)}, ..., and {(xn-2,yn-2),(xn-1,yn-1), (xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

zbuffer Pointer to the image that holds the Z buffer.

x Pointer to array of X coordinates of the points.

y Pointer to array of Y coordinates of the points.

z Pointer to array of Z coordinates of the points.

npoints Number of points in the arrays. npoints must be a multiple of 3.

c Pointer to array of colors of the points.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsFillTriangleSet_8(3MLIB), mllib_GraphicsFillTriangleSet_A_8(3MLIB), mllib_GraphicsFillTriangleSet_AG_8(3MLIB), mllib_GraphicsFillTriangleSet_AZ_8(3MLIB), mllib_GraphicsFillTriangleSet_G_8(3MLIB), mllib_GraphicsFillTriangleSet_GZ_8(3MLIB), mllib_GraphicsFillTriangleSet_X_8(3MLIB), mllib_GraphicsFillTriangleSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsFillTriangleSet_AZ_8(3MLIB)

NAME	mllib_GraphicsFillTriangleSet_AZ_8, mllib_GraphicsFillTriangleSet_AZ_32 – draw filled triangle set with antialiasing and Z buffering, where each member can have different vertices														
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_GraphicsFillTriangleSet_AZ_8(mllib_image *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *z, mllib_s32 npoints, mllib_s32 c); mllib_status mllib_GraphicsFillTriangleSet_AZ_32(mllib_image *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *z, mllib_s32 npoints, mllib_s32 c);</pre>														
DESCRIPTION	Each of these functions draws set of triangles with vertices at {(x1,y1),(x2,y2), (x3,y3)}, {(x4,y4),(x5,y5),(x6,y6)}, ..., and {(xn-2,yn-2),(xn-1,yn-1),(xn,yn)}.														
PARAMETERS	Each of the functions takes the following arguments: <table border="0" style="margin-left: 20px;"> <tr> <td style="padding-right: 20px;"><i>buffer</i></td> <td>Pointer to the image into which the function is drawing.</td> </tr> <tr> <td style="padding-right: 20px;"><i>zbuffer</i></td> <td>Pointer to the image that holds the Z buffer.</td> </tr> <tr> <td style="padding-right: 20px;"><i>x</i></td> <td>Pointer to array of X coordinates of the points.</td> </tr> <tr> <td style="padding-right: 20px;"><i>y</i></td> <td>Pointer to array of Y coordinates of the points.</td> </tr> <tr> <td style="padding-right: 20px;"><i>z</i></td> <td>Pointer to array of Z coordinates of the points.</td> </tr> <tr> <td style="padding-right: 20px;"><i>npoints</i></td> <td>Number of points in the arrays. npoints must be a multiple of 3.</td> </tr> <tr> <td style="padding-right: 20px;"><i>c</i></td> <td>Color used in the drawing.</td> </tr> </table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>zbuffer</i>	Pointer to the image that holds the Z buffer.	<i>x</i>	Pointer to array of X coordinates of the points.	<i>y</i>	Pointer to array of Y coordinates of the points.	<i>z</i>	Pointer to array of Z coordinates of the points.	<i>npoints</i>	Number of points in the arrays. npoints must be a multiple of 3.	<i>c</i>	Color used in the drawing.
<i>buffer</i>	Pointer to the image into which the function is drawing.														
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.														
<i>x</i>	Pointer to array of X coordinates of the points.														
<i>y</i>	Pointer to array of Y coordinates of the points.														
<i>z</i>	Pointer to array of Z coordinates of the points.														
<i>npoints</i>	Number of points in the arrays. npoints must be a multiple of 3.														
<i>c</i>	Color used in the drawing.														
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.														
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1" style="margin-left: 20px; width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe								
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														
MT-Level	MT-Safe														
SEE ALSO	mllib_GraphicsFillTriangleSet_8(3MLIB), mllib_GraphicsFillTriangleSet_A_8(3MLIB), mllib_GraphicsFillTriangleSet_AG_8(3MLIB), mllib_GraphicsFillTriangleSet_AGZ_8(3MLIB), mllib_GraphicsFillTriangleSet_G_8(3MLIB), mllib_GraphicsFillTriangleSet_GZ_8(3MLIB), mllib_GraphicsFillTriangleSet_X_8(3MLIB), mllib_GraphicsFillTriangleSet_Z_8(3MLIB), attributes(5)														

mllib_GraphicsFillTriangleSet_G_8(3MLIB)

NAME mllib_GraphicsFillTriangleSet_G_8, mllib_GraphicsFillTriangleSet_G_32 – draw filled triangle set with Gouraud shading where each member can have different vertices

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillTriangleSet_G_8(mllib_image *buffer,
        const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, const
        mllib_s32 *c);

mllib_status mllib_GraphicsFillTriangleSet_G_32(mllib_image *buffer,
        const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, const
        mllib_s32 *c);
```

DESCRIPTION Each of these functions draws set of triangles with vertices at {(x1,y1),(x2,y2), (x3,y3)}, {(x4,y4),(x5,y5),(x6,y6)}, ..., and {(xn-2,yn-2),(xn-1,yn-1),(xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

x Pointer to array of X coordinates of the points.

y Pointer to array of Y coordinates of the points.

npoints Number of points in the arrays. npoints must be a multiple of 3.

c Pointer to array of colors of the points.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsFillTriangleSet_8(3MLIB), mllib_GraphicsFillTriangleSet_A_8(3MLIB), mllib_GraphicsFillTriangleSet_AG_8(3MLIB), mllib_GraphicsFillTriangleSet_AGZ_8(3MLIB), mllib_GraphicsFillTriangleSet_AZ_8(3MLIB), mllib_GraphicsFillTriangleSet_GZ_8(3MLIB), mllib_GraphicsFillTriangleSet_X_8(3MLIB), mllib_GraphicsFillTriangleSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsFillTriangleSet_GZ_8(3MLIB)

NAME	mllib_GraphicsFillTriangleSet_GZ_8, mllib_GraphicsFillTriangleSet_GZ_32 – draw filled triangle set with Gouraud shading and Z buffering, where each member can have different vertices														
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_GraphicsFillTriangleSet_GZ_8(mllib_image *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c); mllib_status mllib_GraphicsFillTriangleSet_GZ_32(mllib_image *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);</pre>														
DESCRIPTION	Each of these functions draws a filled set of triangles with vertices at {(x1,y1), (x2,y2),(x3,y3)}, {(x4,y4),(x5,y5),(x6,y6)}, ..., and {(xn-2,yn-2),(xn-1,yn-1), (xn,yn)}.														
PARAMETERS	Each of the functions takes the following arguments: <table border="0" style="margin-left: 20px;"> <tr> <td style="padding-right: 20px;"><i>buffer</i></td> <td>Pointer to the image into which the function is drawing.</td> </tr> <tr> <td><i>zbuffer</i></td> <td>Pointer to the image that holds the Z buffer.</td> </tr> <tr> <td><i>x</i></td> <td>Pointer to array of X coordinates of the points.</td> </tr> <tr> <td><i>y</i></td> <td>Pointer to array of Y coordinates of the points.</td> </tr> <tr> <td><i>z</i></td> <td>Pointer to array of Z coordinates of the points.</td> </tr> <tr> <td><i>npoints</i></td> <td>Number of points in the arrays. npoints must be a multiple of 3.</td> </tr> <tr> <td><i>c</i></td> <td>Pointer to array of colors of the points.</td> </tr> </table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>zbuffer</i>	Pointer to the image that holds the Z buffer.	<i>x</i>	Pointer to array of X coordinates of the points.	<i>y</i>	Pointer to array of Y coordinates of the points.	<i>z</i>	Pointer to array of Z coordinates of the points.	<i>npoints</i>	Number of points in the arrays. npoints must be a multiple of 3.	<i>c</i>	Pointer to array of colors of the points.
<i>buffer</i>	Pointer to the image into which the function is drawing.														
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.														
<i>x</i>	Pointer to array of X coordinates of the points.														
<i>y</i>	Pointer to array of Y coordinates of the points.														
<i>z</i>	Pointer to array of Z coordinates of the points.														
<i>npoints</i>	Number of points in the arrays. npoints must be a multiple of 3.														
<i>c</i>	Pointer to array of colors of the points.														
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.														
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1" style="margin-left: 20px; width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe								
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														
MT-Level	MT-Safe														
SEE ALSO	<p>mllib_GraphicsFillTriangleSet_8(3MLIB), mllib_GraphicsFillTriangleSet_A_8(3MLIB), mllib_GraphicsFillTriangleSet_AG_8(3MLIB), mllib_GraphicsFillTriangleSet_AGZ_8(3MLIB), mllib_GraphicsFillTriangleSet_AZ_8(3MLIB), mllib_GraphicsFillTriangleSet_G_8(3MLIB), mllib_GraphicsFillTriangleSet_X_8(3MLIB), mllib_GraphicsFillTriangleSet_Z_8(3MLIB), attributes(5)</p>														

mllib_GraphicsFillTriangleSet_X_8(3MLIB)

NAME mllib_GraphicsFillTriangleSet_X_8, mllib_GraphicsFillTriangleSet_X_32 – draw filled triangle set in Xor mode where each member can have different vertices

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillTriangleSet_X_8(mllib_image *buffer,
        const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
        mllib_s32 c, mllib_s32 c2);

mllib_status mllib_GraphicsFillTriangleSet_X_32(mllib_image *buffer,
        const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
        mllib_s32 c, mllib_s32 c2);
```

DESCRIPTION Each of these functions draws set of triangles with vertices at {(x1,y1),(x2,y2), (x3,y3)}, {(x4,y4),(x5,y5),(x6,y6)}, ..., and {(xn-2,yn-2),(xn-1,yn-1),(xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

x Pointer to array of X coordinates of the points.

y Pointer to array of Y coordinates of the points.

npoints Number of points in the arrays. *npoints* must be a multiple of 3.

c Color used in the drawing.

c2 Alternation color.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsFillTriangleSet_8(3MLIB), mllib_GraphicsFillTriangleSet_A_8(3MLIB), mllib_GraphicsFillTriangleSet_AG_8(3MLIB), mllib_GraphicsFillTriangleSet_AGZ_8(3MLIB), mllib_GraphicsFillTriangleSet_AZ_8(3MLIB), mllib_GraphicsFillTriangleSet_G_8(3MLIB), mllib_GraphicsFillTriangleSet_GZ_8(3MLIB), mllib_GraphicsFillTriangleSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsFillTriangleSet_Z_8(3MLIB)

NAME	mllib_GraphicsFillTriangleSet_Z_8, mllib_GraphicsFillTriangleSet_Z_32 – draw filled triangle set with Z buffering where each member can have different vertices														
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsFillTriangleSet_Z_8(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, const mllib_s16 *<i>z</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>); mllib_status mllib_GraphicsFillTriangleSet_Z_32(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, const mllib_s16 *<i>x</i>, const mllib_s16 *<i>y</i>, const mllib_s16 *<i>z</i>, mllib_s32 <i>npoints</i>, mllib_s32 <i>c</i>);</pre>														
DESCRIPTION	Each of these functions draws set of triangles with vertices at {(x1,y1),(x2,y2), (x3,y3)}, {(x4,y4),(x5,y5),(x6,y6)}, ..., and {(xn-2,yn-2),(xn-1,yn-1),(xn,yn)}.														
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>zbuffer</i></td><td>Pointer to the image that holds the Z buffer.</td></tr><tr><td><i>x</i></td><td>Pointer to array of X coordinates of the points.</td></tr><tr><td><i>y</i></td><td>Pointer to array of Y coordinates of the points.</td></tr><tr><td><i>z</i></td><td>Pointer to array of Z coordinates of the points.</td></tr><tr><td><i>npoints</i></td><td>Number of points in the arrays. npoints must be a multiple of 3.</td></tr><tr><td><i>c</i></td><td>Color used in the drawing.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>zbuffer</i>	Pointer to the image that holds the Z buffer.	<i>x</i>	Pointer to array of X coordinates of the points.	<i>y</i>	Pointer to array of Y coordinates of the points.	<i>z</i>	Pointer to array of Z coordinates of the points.	<i>npoints</i>	Number of points in the arrays. npoints must be a multiple of 3.	<i>c</i>	Color used in the drawing.
<i>buffer</i>	Pointer to the image into which the function is drawing.														
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.														
<i>x</i>	Pointer to array of X coordinates of the points.														
<i>y</i>	Pointer to array of Y coordinates of the points.														
<i>z</i>	Pointer to array of Z coordinates of the points.														
<i>npoints</i>	Number of points in the arrays. npoints must be a multiple of 3.														
<i>c</i>	Color used in the drawing.														
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.														
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe								
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														
MT-Level	MT-Safe														
SEE ALSO	mllib_GraphicsFillTriangleSet_8(3MLIB), mllib_GraphicsFillTriangleSet_A_8(3MLIB), mllib_GraphicsFillTriangleSet_AG_8(3MLIB), mllib_GraphicsFillTriangleSet_AGZ_8(3MLIB), mllib_GraphicsFillTriangleSet_AZ_8(3MLIB), mllib_GraphicsFillTriangleSet_G_8(3MLIB), mllib_GraphicsFillTriangleSet_GZ_8(3MLIB), mllib_GraphicsFillTriangleSet_X_8(3MLIB), attributes(5)														

mllib_GraphicsFillTriangleStripSet_8(3MLIB)

NAME mllib_GraphicsFillTriangleStripSet_8, mllib_GraphicsFillTriangleStripSet_32 – draw filled triangle set where the first side of each member is common to the second side of the previous member

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillTriangleStripSet_8(mllib_image
    *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
    mllib_s32 c);

mllib_status mllib_GraphicsFillTriangleStripSet_32(mllib_image
    *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
    mllib_s32 c);
```

DESCRIPTION Each of these functions draws a filled set of triangles with vertices at {(x1,y1), (x2,y2),(x3,y3)}, {(x2,y2),(x3,y3),(x4,y4)}, ..., and {(xn-2,yn-2),(xn-1,yn-1), (xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.
x Pointer to array of X coordinates of the points.
y Pointer to array of Y coordinates of the points.
npoints Number of points in the arrays.
c Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsFillTriangleStripSet_A_8(3MLIB), mllib_GraphicsFillTriangleStripSet_AG_8(3MLIB), mllib_GraphicsFillTriangleStripSet_AGZ(3MLIB), mllib_GraphicsFillTriangleStripSet_AZ_8(3MLIB), mllib_GraphicsFillTriangleStripSet_G_8(3MLIB), mllib_GraphicsFillTriangleStripSet_GZ_8(3MLIB), mllib_GraphicsFillTriangleStripSet_X_8(3MLIB), mllib_GraphicsFillTriangleStripSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsFillTriangleStripSet_A_8(3MLIB)

NAME	mllib_GraphicsFillTriangleStripSet_A_8, mllib_GraphicsFillTriangleStripSet_A_32 – draw filled triangle set with antialiasing where the first side of each member is common to the second side of the previous member						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsFillTriangleStripSet_A_8(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c); mllib_status mllib_GraphicsFillTriangleStripSet_A_32(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c);</pre>						
DESCRIPTION	Each of these functions draws a filled set of triangles with vertices at {(x1,y1), (x2,y2),(x3,y3)}, {(x2,y2),(x3,y3),(x4,y4)}, ..., and {(xn-2,yn-2),(xn-1,yn-1), (xn,yn)}.						
PARAMETERS	Each of the functions takes the following arguments: <i>buffer</i> Pointer to the image into which the function is drawing. <i>x</i> Pointer to array of X coordinates of the points. <i>y</i> Pointer to array of Y coordinates of the points. <i>npoints</i> Number of points in the arrays. <i>c</i> Color used in the drawing.						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_GraphicsFillTriangleStripSet_8(3MLIB), mllib_GraphicsFillTriangleStripSet_AG_8(3MLIB), mllib_GraphicsFillTriangleStripSet_AGZ(3MLIB), mllib_GraphicsFillTriangleStripSet_AZ_8(3MLIB), mllib_GraphicsFillTriangleStripSet_G_8(3MLIB), mllib_GraphicsFillTriangleStripSet_GZ_8(3MLIB), mllib_GraphicsFillTriangleStripSet_X_8(3MLIB), mllib_GraphicsFillTriangleStripSet_Z_8(3MLIB), attributes(5)						

mllib_GraphicsFillTriangleStripSet_AG_8(3MLIB)

NAME mllib_GraphicsFillTriangleStripSet_AG_8, mllib_GraphicsFillTriangleStripSet_AG_32 – draw filled triangle set with antialiasing and gouraud shading, where the first side of each member is common to the second side of the previous member

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillTriangleStripSet_AG_8(mllib_image
    *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
    const mllib_s32 *c);

mllib_status mllib_GraphicsFillTriangleStripSet_AG_32(mllib_image
    *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints,
    const mllib_s32 *c);
```

DESCRIPTION Each of these functions draws a filled set of triangles with vertices at {(x1,y1), (x2,y2),(x3,y3)}, {(x2,y2),(x3,y3),(x4,y4)}, ..., and {(xn-2,yn-2),(xn-1,yn-1), (xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.
x Pointer to array of X coordinates of the points.
y Pointer to array of Y coordinates of the points.
npoints Number of points in the arrays.
c Pointer to array of colors of the points.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsFillTriangleStripSet_8(3MLIB), mllib_GraphicsFillTriangleStripSet_A_8(3MLIB), mllib_GraphicsFillTriangleStripSet_AGZ(3MLIB), mllib_GraphicsFillTriangleStripSet_AZ_8(3MLIB), mllib_GraphicsFillTriangleStripSet_G_8(3MLIB), mllib_GraphicsFillTriangleStripSet_GZ_8(3MLIB), mllib_GraphicsFillTriangleStripSet_X_8(3MLIB), mllib_GraphicsFillTriangleStripSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsFillTriangleStripSet_AGZ(3MLIB)

NAME mllib_GraphicsFillTriangleStripSet_AGZ, mllib_GraphicsFillTriangleStripSet_AGZ_8, mllib_GraphicsFillTriangleStripSet_AGZ_32 – draw filled triangle set with antialiasing, Gouraud shading, and Z buffering, where the first side of each member is common to the second side of the previous member

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillTriangleStripSet_AGZ_8(mllib_image
    *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16
    *y, const mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);

mllib_status mllib_GraphicsFillTriangleStripSet_AGZ_32(mllib_image
    *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16
    *y, const mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);
```

DESCRIPTION Each of these functions draws a filled set of triangles with vertices at {(x1,y1), (x2,y2),(x3,y3)}, {(x2,y2),(x3,y3),(x4,y4)}, ..., and {(xn-2,yn-2),(xn-1,yn-1), (xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

zbuffer Pointer to the image that holds the Z buffer.

x Pointer to array of X coordinates of the points.

y Pointer to array of Y coordinates of the points.

z Pointer to array of Z coordinates of the points.

npoints Number of points in the arrays.

c Pointer to array of colors of the points.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsFillTriangleStripSet_8(3MLIB), mllib_GraphicsFillTriangleStripSet_A_8(3MLIB), mllib_GraphicsFillTriangleStripSet_AG_8(3MLIB), mllib_GraphicsFillTriangleStripSet_AZ_8(3MLIB), mllib_GraphicsFillTriangleStripSet_G_8(3MLIB), mllib_GraphicsFillTriangleStripSet_GZ_8(3MLIB), mllib_GraphicsFillTriangleStripSet_X_8(3MLIB), mllib_GraphicsFillTriangleStripSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsFillTriangleStripSet_AZ_8(3MLIB)

NAME mllib_GraphicsFillTriangleStripSet_AZ_8, mllib_GraphicsFillTriangleStripSet_AZ_32 – draw filled triangle set with antialiasing and Z buffering, where the first side of each member is common to the second side of the previous member

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillTriangleStripSet_AZ_8(mllib_image
    *buffer, mllib_image *zbuffer, const mlib_s16 *x, const mlib_s16
    *y, const mlib_s16 *z, mlib_s32 npoints, mlib_s32 c);

mllib_status mllib_GraphicsFillTriangleStripSet_AZ_32(mllib_image
    *buffer, mllib_image *zbuffer, const mlib_s16 *x, const mlib_s16
    *y, const mlib_s16 *z, mlib_s32 npoints, mlib_s32 c);
```

DESCRIPTION Each of these functions draws a filled set of triangles with vertices at {(x1,y1), (x2,y2),(x3,y3)}, {(x2,y2),(x3,y3),(x4,y4)}, ..., and {(xn-2,yn-2),(xn-1,yn-1), (xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

zbuffer Pointer to the image that holds the Z buffer.

x Pointer to array of X coordinates of the points.

y Pointer to array of Y coordinates of the points.

z Pointer to array of Z coordinates of the points.

npoints Number of points in the arrays.

c Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsFillTriangleStripSet_8(3MLIB), mllib_GraphicsFillTriangleStripSet_A_8(3MLIB), mllib_GraphicsFillTriangleStripSet_AG_8(3MLIB), mllib_GraphicsFillTriangleStripSet_AGZ(3MLIB), mllib_GraphicsFillTriangleStripSet_G_8(3MLIB), mllib_GraphicsFillTriangleStripSet_GZ_8(3MLIB), mllib_GraphicsFillTriangleStripSet_X_8(3MLIB), mllib_GraphicsFillTriangleStripSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsFillTriangleStripSet_G_8(3MLIB)

NAME	mllib_GraphicsFillTriangleStripSet_G_8, mllib_GraphicsFillTriangleStripSet_G_32 – draw filled triangle set with Gouraud shading where the first side of each member is common to the second side of the previous member						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_GraphicsFillTriangleStripSet_G_8(mllib_image *buffer, const mlib_s16 *x, const mlib_s16 *y, mlib_s32 npoints, const mlib_s32 *c); mllib_status mllib_GraphicsFillTriangleStripSet_G_32(mllib_image *buffer, const mlib_s16 *x, const mlib_s16 *y, mlib_s32 npoints, const mlib_s32 *c);</pre>						
DESCRIPTION	Each of these functions draws a set of triangles with vertices at {(x1,y1),(x2,y2),(x3,y3)}, {(x2,y2),(x3,y3),(x4,y4)}, ..., and {(xn-2,yn-2),(xn-1,yn-1),(xn,yn)}.						
PARAMETERS	Each of the functions takes the following arguments: <p><i>buffer</i> Pointer to the image into which the function is drawing.</p> <p><i>x</i> Pointer to array of X coordinates of the points.</p> <p><i>y</i> Pointer to array of Y coordinates of the points.</p> <p><i>npoints</i> Number of points in the arrays.</p> <p><i>c</i> Pointer to array of colors of the points.</p>						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<p>mllib_GraphicsFillTriangleStripSet_8(3MLIB), mllib_GraphicsFillTriangleStripSet_A_8(3MLIB), mllib_GraphicsFillTriangleStripSet_AG_8(3MLIB), mllib_GraphicsFillTriangleStripSet_AGZ(3MLIB), mllib_GraphicsFillTriangleStripSet_AZ_8(3MLIB), mllib_GraphicsFillTriangleStripSet_GZ_8(3MLIB), mllib_GraphicsFillTriangleStripSet_X_8(3MLIB), mllib_GraphicsFillTriangleStripSet_Z_8(3MLIB), attributes(5)</p>						

mllib_GraphicsFillTriangleStripSet_GZ_8(3MLIB)

NAME mllib_GraphicsFillTriangleStripSet_GZ_8, mllib_GraphicsFillTriangleStripSet_GZ_32 – draw filled triangle set with Gouraud shading and Z buffering, where the first side of each member is common to the second side of the previous member

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillTriangleStripSet_GZ_8(mllib_image
    *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16
    *y, const mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);

mllib_status mllib_GraphicsFillTriangleStripSet_GZ_32(mllib_image
    *buffer, mllib_image *zbuffer, const mllib_s16 *x, const mllib_s16
    *y, const mllib_s16 *z, mllib_s32 npoints, const mllib_s32 *c);
```

DESCRIPTION Each of these functions draws a filled set of triangles with vertices at {(x1,y1), (x2,y2),(x3,y3)}, {(x2,y2),(x3,y3),(x4,y4)}, ..., and {(xn-2,yn-2),(xn-1,yn-1), (xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

zbuffer Pointer to the image that holds the Z buffer.

x Pointer to array of X coordinates of the points.

y Pointer to array of Y coordinates of the points.

z Pointer to array of Z coordinates of the points.

npoints Number of points in the arrays.

c Pointer to array of colors of the points.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsFillTriangleStripSet_8(3MLIB), mllib_GraphicsFillTriangleStripSet_A_8(3MLIB), mllib_GraphicsFillTriangleStripSet_AG_8(3MLIB), mllib_GraphicsFillTriangleStripSet_AGZ(3MLIB), mllib_GraphicsFillTriangleStripSet_AZ_8(3MLIB), mllib_GraphicsFillTriangleStripSet_G_8(3MLIB), mllib_GraphicsFillTriangleStripSet_X_8(3MLIB), mllib_GraphicsFillTriangleStripSet_Z_8(3MLIB), attributes(5)

mllib_GraphicsFillTriangleStripSet_X_8(3MLIB)

NAME	mllib_GraphicsFillTriangleStripSet_X_8, mllib_GraphicsFillTriangleStripSet_X_32 – draw filled triangle set in Xor mode where the first side of each member is common to the second side of the previous member						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsFillTriangleStripSet_X_8(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c, mllib_s32 c2); mllib_status mllib_GraphicsFillTriangleStripSet_X_32(mllib_image *buffer, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 npoints, mllib_s32 c, mllib_s32 c2);</pre>						
DESCRIPTION	Each of these functions draws a filled set of triangles with vertices at {(x1,y1), (x2,y2),(x3,y3)}, {(x2,y2),(x3,y3),(x4,y4)}, ..., and {(xn-2,yn-2),(xn-1,yn-1), (xn,yn)}.						
PARAMETERS	Each of the functions takes the following arguments: <i>buffer</i> Pointer to the image into which the function is drawing. <i>x</i> Pointer to array of X coordinates of the points. <i>y</i> Pointer to array of Y coordinates of the points. <i>npoints</i> Number of points in the arrays. <i>c</i> Color used in the drawing. <i>c2</i> Alternation color.						
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_GraphicsFillTriangleStripSet_8(3MLIB), mllib_GraphicsFillTriangleStripSet_A_8(3MLIB), mllib_GraphicsFillTriangleStripSet_AG_8(3MLIB), mllib_GraphicsFillTriangleStripSet_AGZ(3MLIB), mllib_GraphicsFillTriangleStripSet_AZ_8(3MLIB), mllib_GraphicsFillTriangleStripSet_G_8(3MLIB), mllib_GraphicsFillTriangleStripSet_GZ_8(3MLIB), mllib_GraphicsFillTriangleStripSet_Z_8(3MLIB), attributes(5)						

mllib_GraphicsFillTriangleStripSet_Z_8(3MLIB)

NAME mllib_GraphicsFillTriangleStripSet_Z_8, mllib_GraphicsFillTriangleStripSet_Z_32 – draw filled triangle set with Z buffering where the first side of each member is common to the second side of the previous member

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_GraphicsFillTriangleStripSet_Z_8(mllib_image
    *buffer, mllib_image *zbuffer, const mlib_s16 *x, const mlib_s16
    *y, const mlib_s16 *z, mlib_s32 npoints, mlib_s32 c);

mllib_status mllib_GraphicsFillTriangleStripSet_Z_32(mllib_image
    *buffer, mllib_image *zbuffer, const mlib_s16 *x, const mlib_s16
    *y, const mlib_s16 *z, mlib_s32 npoints, mlib_s32 c);
```

DESCRIPTION Each of these functions draws a filled set of triangles with vertices at {(x1,y1), (x2,y2),(x3,y3)}, {(x2,y2),(x3,y3),(x4,y4)}, ..., and {(xn-2,yn-2),(xn-1,yn-1), (xn,yn)}.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

zbuffer Pointer to the image that holds the Z buffer.

x Pointer to array of X coordinates of the points.

y Pointer to array of Y coordinates of the points.

z Pointer to array of Z coordinates of the points.

npoints Number of points in the arrays.

c Color used in the drawing.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_GraphicsFillTriangleStripSet_8(3MLIB), mllib_GraphicsFillTriangleStripSet_A_8(3MLIB), mllib_GraphicsFillTriangleStripSet_AG_8(3MLIB), mllib_GraphicsFillTriangleStripSet_AGZ(3MLIB), mllib_GraphicsFillTriangleStripSet_AZ_8(3MLIB), mllib_GraphicsFillTriangleStripSet_G_8(3MLIB), mllib_GraphicsFillTriangleStripSet_GZ_8(3MLIB), mllib_GraphicsFillTriangleStripSet_X_8(3MLIB), attributes(5)

mllib_GraphicsFillTriangle_X_8(3MLIB)

NAME	mllib_GraphicsFillTriangle_X_8, mllib_GraphicsFillTriangle_X_32 – draw filled triangle in Xor mode																		
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsFillTriangle_X_8(mllib_image *<i>buffer</i>, mllib_s16 <i>x1</i>, mllib_s16 <i>y1</i>, mllib_s16 <i>x2</i>, mllib_s16 <i>y2</i>, mllib_s16 <i>x3</i>, mllib_s16 <i>y3</i>, mllib_s32 <i>c</i>, mllib_s32 <i>c2</i>); mllib_status mllib_GraphicsFillTriangle_X_32(mllib_image *<i>buffer</i>, mllib_s16 <i>x1</i>, mllib_s16 <i>y1</i>, mllib_s16 <i>x2</i>, mllib_s16 <i>y2</i>, mllib_s16 <i>x3</i>, mllib_s16 <i>y3</i>, mllib_s32 <i>c</i>, mllib_s32 <i>c2</i>);</pre>																		
DESCRIPTION	Each of these functions draws a filled triangle with the vertices at (x1,y1), (x2,y2), and (x3,y3).																		
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>buffer</i></td><td>Pointer to the image into which the function is drawing.</td></tr><tr><td><i>x1</i></td><td>X coordinate of the first vertex.</td></tr><tr><td><i>y1</i></td><td>Y coordinate of the first vertex.</td></tr><tr><td><i>x2</i></td><td>X coordinate of the second vertex.</td></tr><tr><td><i>y2</i></td><td>Y coordinate of the second vertex.</td></tr><tr><td><i>x3</i></td><td>X coordinate of the third vertex.</td></tr><tr><td><i>y3</i></td><td>Y coordinate of the third vertex.</td></tr><tr><td><i>c</i></td><td>Color used in the drawing.</td></tr><tr><td><i>c2</i></td><td>Alternation color.</td></tr></table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>x1</i>	X coordinate of the first vertex.	<i>y1</i>	Y coordinate of the first vertex.	<i>x2</i>	X coordinate of the second vertex.	<i>y2</i>	Y coordinate of the second vertex.	<i>x3</i>	X coordinate of the third vertex.	<i>y3</i>	Y coordinate of the third vertex.	<i>c</i>	Color used in the drawing.	<i>c2</i>	Alternation color.
<i>buffer</i>	Pointer to the image into which the function is drawing.																		
<i>x1</i>	X coordinate of the first vertex.																		
<i>y1</i>	Y coordinate of the first vertex.																		
<i>x2</i>	X coordinate of the second vertex.																		
<i>y2</i>	Y coordinate of the second vertex.																		
<i>x3</i>	X coordinate of the third vertex.																		
<i>y3</i>	Y coordinate of the third vertex.																		
<i>c</i>	Color used in the drawing.																		
<i>c2</i>	Alternation color.																		
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.																		
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe												
ATTRIBUTE TYPE	ATTRIBUTE VALUE																		
Interface Stability	Evolving																		
MT-Level	MT-Safe																		
SEE ALSO	<code>mllib_GraphicsFillTriangle_8(3MLIB)</code> , <code>mllib_GraphicsFillTriangle_A_8(3MLIB)</code> , <code>mllib_GraphicsFillTriangle_AG_8(3MLIB)</code> , <code>mllib_GraphicsFillTriangle_AGZ_8(3MLIB)</code> ,																		

`mllib_GraphicsFillTriangle_X_8(3MLIB)`

```
mllib_GraphicsFillTriangle_AZ_8(3MLIB),  
mllib_GraphicsFillTriangle_G_8(3MLIB),  
mllib_GraphicsFillTriangle_GZ_8(3MLIB),  
mllib_GraphicsFillTriangle_Z_8(3MLIB), attributes(5)
```

mllib_GraphicsFillTriangle_Z_8(3MLIB)

NAME	mllib_GraphicsFillTriangle_Z_8, mllib_GraphicsFillTriangle_Z_32 – draw filled triangle with Z buffering																								
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_GraphicsFillTriangle_Z_8(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, mllib_s16 <i>x1</i>, mllib_s16 <i>y1</i>, mllib_s16 <i>z1</i>, mllib_s16 <i>x2</i>, mllib_s16 <i>y2</i>, mllib_s16 <i>z2</i>, mllib_s16 <i>x3</i>, mllib_s16 <i>y3</i>, mllib_s16 <i>z3</i>, mllib_s32 <i>c</i>); mllib_status mllib_GraphicsFillTriangle_Z_32(mllib_image *<i>buffer</i>, mllib_image *<i>zbuffer</i>, mllib_s16 <i>x1</i>, mllib_s16 <i>y1</i>, mllib_s16 <i>z1</i>, mllib_s16 <i>x2</i>, mllib_s16 <i>y2</i>, mllib_s16 <i>z2</i>, mllib_s16 <i>x3</i>, mllib_s16 <i>y3</i>, mllib_s16 <i>z3</i>, mllib_s32 <i>c</i>);</pre>																								
DESCRIPTION	Each of these functions draws a filled triangle with the vertices at (x1,y1), (x2,y2), and (x3,y3).																								
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <table border="0"> <tr> <td style="padding-right: 20px;"><i>buffer</i></td> <td>Pointer to the image into which the function is drawing.</td> </tr> <tr> <td><i>zbuffer</i></td> <td>Pointer to the image that holds the Z buffer.</td> </tr> <tr> <td><i>x1</i></td> <td>X coordinate of the first vertex.</td> </tr> <tr> <td><i>y1</i></td> <td>Y coordinate of the first vertex.</td> </tr> <tr> <td><i>z1</i></td> <td>Z coordinate of the first vertex.</td> </tr> <tr> <td><i>x2</i></td> <td>X coordinate of the second vertex.</td> </tr> <tr> <td><i>y2</i></td> <td>Y coordinate of the second vertex.</td> </tr> <tr> <td><i>z2</i></td> <td>Z coordinate of the second vertex.</td> </tr> <tr> <td><i>x3</i></td> <td>X coordinate of the third vertex.</td> </tr> <tr> <td><i>y3</i></td> <td>Y coordinate of the third vertex.</td> </tr> <tr> <td><i>z3</i></td> <td>Z coordinate of the third vertex.</td> </tr> <tr> <td><i>c</i></td> <td>Color used in the drawing.</td> </tr> </table>	<i>buffer</i>	Pointer to the image into which the function is drawing.	<i>zbuffer</i>	Pointer to the image that holds the Z buffer.	<i>x1</i>	X coordinate of the first vertex.	<i>y1</i>	Y coordinate of the first vertex.	<i>z1</i>	Z coordinate of the first vertex.	<i>x2</i>	X coordinate of the second vertex.	<i>y2</i>	Y coordinate of the second vertex.	<i>z2</i>	Z coordinate of the second vertex.	<i>x3</i>	X coordinate of the third vertex.	<i>y3</i>	Y coordinate of the third vertex.	<i>z3</i>	Z coordinate of the third vertex.	<i>c</i>	Color used in the drawing.
<i>buffer</i>	Pointer to the image into which the function is drawing.																								
<i>zbuffer</i>	Pointer to the image that holds the Z buffer.																								
<i>x1</i>	X coordinate of the first vertex.																								
<i>y1</i>	Y coordinate of the first vertex.																								
<i>z1</i>	Z coordinate of the first vertex.																								
<i>x2</i>	X coordinate of the second vertex.																								
<i>y2</i>	Y coordinate of the second vertex.																								
<i>z2</i>	Z coordinate of the second vertex.																								
<i>x3</i>	X coordinate of the third vertex.																								
<i>y3</i>	Y coordinate of the third vertex.																								
<i>z3</i>	Z coordinate of the third vertex.																								
<i>c</i>	Color used in the drawing.																								
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.																								
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:																								
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe																		
ATTRIBUTE TYPE	ATTRIBUTE VALUE																								
Interface Stability	Evolving																								
MT-Level	MT-Safe																								

`mllib_GraphicsFillTriangle_Z_8(3MLIB)`

SEE ALSO `mllib_GraphicsFillTriangle_8(3MLIB)`,
`mllib_GraphicsFillTriangle_A_8(3MLIB)`,
`mllib_GraphicsFillTriangle_AG_8(3MLIB)`,
`mllib_GraphicsFillTriangle_AGZ_8(3MLIB)`,
`mllib_GraphicsFillTriangle_AZ_8(3MLIB)`,
`mllib_GraphicsFillTriangle_G_8(3MLIB)`,
`mllib_GraphicsFillTriangle_GZ_8(3MLIB)`,
`mllib_GraphicsFillTriangle_X_8(3MLIB)`, `attributes(5)`

mllib_GraphicsFloodFill_8(3MLIB)

NAME mllib_GraphicsFloodFill_8, mllib_GraphicsFloodFill_32 – flood fill

SYNOPSIS cc [*flag...*] *file...* -lmlib [*library...*]
#include <mllib.h>

```
mllib_status mllib_GraphicsFloodFill_8(mllib_image *buffer, mllib_s16
    x, mllib_s16 y, mllib_s32 c, mllib_s32 c2);
mllib_status mllib_GraphicsFloodFill_32(mllib_image *buffer, mllib_s16
    x, mllib_s16 y, mllib_s32 c, mllib_s32 c2);
```

DESCRIPTION Each of these functions performs flood fill.

PARAMETERS Each of the functions takes the following arguments:

buffer Pointer to the image into which the function is drawing.

x X coordinate of the starting point.

y Y coordinate of the starting point.

c Color used in the drawing.

c2 Color that defines the filling interior.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO attributes(5)

NAME mllib_ImageAbs – computes the absolute value of the image pixels

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageAbs(mllib_image *dst, const mllib_image *src);
```

DESCRIPTION The mllib_ImageAbs() function computes the absolute value of the image pixels. It uses the following equation:

$$dst[x][y][i] = |src[x][y][i]|$$

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_ImageAbs_Fp\(3MLIB\)](#), [mllib_ImageAbs_Fp_Inp\(3MLIB\)](#), [mllib_ImageAbs_Inp\(3MLIB\)](#), [attributes\(5\)](#)

mllib_ImageAbs_Fp(3MLIB)

NAME | mllib_ImageAbs_Fp – computes the absolute value of the image pixels

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageAbs_Fp(mllib_image *dst, const mllib_image
    *src);
```

DESCRIPTION | The mllib_ImageAbs_Fp() function computes the floating-point absolute value of the image pixels.

It uses the following equation:

$$dst[x][y][i] = |src[x][y][i]|$$

PARAMETERS | The function takes the following arguments:

dst | Pointer to destination image.

src | Pointer to source image.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_ImageAbs\(3MLIB\)](#), [mllib_ImageAbs_Fp_Inp\(3MLIB\)](#), [mllib_ImageAbs_Inp\(3MLIB\)](#), [attributes\(5\)](#)

mllib_ImageAbs_Fp_Inp(3MLIB)

NAME mllib_ImageAbs_Fp_Inp – computes the absolute value of the image pixels

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageAbs_Fp_Inp(mllib_image *srcdst);
```

DESCRIPTION The mllib_ImageAbs_Fp_Inp() function computes the floating-point absolute value of the image pixels, in place.

It uses the following equation:

$$\text{srcdst}[x][y][i] = |\text{srcdst}[x][y][i]|$$

PARAMETERS The function takes the following arguments:

srcdst Pointer to source and destination image.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_ImageAbs\(3MLIB\)](#), [mllib_ImageAbs_Fp\(3MLIB\)](#), [mllib_ImageAbs_Inp\(3MLIB\)](#), [attributes\(5\)](#)

mllib_ImageAbs_Inp(3MLIB)

NAME	mllib_ImageAbs_Inp – computes the absolute value of the image pixels, in place
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageAbs_Inp(mllib_image *srcdst);</pre>
DESCRIPTION	<p>The <code>mllib_ImageAbs_Inp()</code> function computes the absolute value of the image pixels in place.</p> <p>It uses the following equation:</p> $\text{srcdst}[x][y][i] = \text{srcdst}[x][y][i] $
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>srcdst</i> Pointer to source and destination image.</p>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_ImageAbs\(3MLIB\)](#), [mllib_ImageAbs_Fp\(3MLIB\)](#), [mllib_ImageAbs_Fp_Inp\(3MLIB\)](#), [attributes\(5\)](#)

- NAME** mlib_ImageAdd – computes the addition of two images on a pixel-by-pixel basis
- SYNOPSIS**

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_ImageAdd(mlib_image dst, const mlib_image *src1,
    const mlib_image *src2);
```
- DESCRIPTION** The `mlib_ImageAdd()` function computes the addition of two images on a pixel-by-pixel basis.
- It uses the following equation:
- $$dst[x][y][i] = src1[x][y][i] + src2[x][y][i]$$
- PARAMETERS** The function takes the following arguments:
- dst* Pointer to destination image.
- src1* Pointer to first source image.
- src2* Pointer to second source image.
- RETURN VALUES** The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.
- ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

- SEE ALSO** `mlib_ImageAdd_Fp(3MLIB)`, `mlib_ImageAdd_Fp_Inp(3MLIB)`, `mlib_ImageAdd_Inp(3MLIB)`, `attributes(5)`

mllib_ImageAdd_Fp(3MLIB)

NAME | mllib_ImageAdd_Fp – computes the addition of two images on a pixel-by-pixel basis

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageAdd_Fp(mllib_image dst, const mllib_image
    *src1, const mllib_image *src2);
```

DESCRIPTION | The `mllib_ImageAdd_Fp()` function computes the addition of two floating-point images on a pixel-by-pixel basis.

It uses the following equation:

$$dst[x][y][i] = src1[x][y][i] + src2[x][y][i]$$

PARAMETERS | The function takes the following arguments:

dst | Pointer to destination image.

src1 | Pointer to first source image.

src2 | Pointer to second source image.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageAdd(3MLIB)`, `mllib_ImageAdd_Fp_Inp(3MLIB)`,
`mllib_ImageAdd_Inp(3MLIB)`, `attributes(5)`

mlib_ImageAdd_Fp_Inp(3MLIB)

NAME | mlib_ImageAdd_Fp_Inp – computes the addition of two images on a pixel-by-pixel basis

SYNOPSIS |

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_ImageAdd_Fp_Inp(mlib_image *src1dst, const
    mlib_image *src2);
```

DESCRIPTION | The `mlib_ImageAdd_Fp_Inp()` function computes the addition of two floating-point images on a pixel-by-pixel basis, in place.

It uses the following equation:

$$src1dst[x][y][i] = src1dst[x][y][i] + src2[x][y][i]$$

PARAMETERS | The function takes the following arguments:

src1dst | Pointer to first source and destination image.

src2 | Pointer to second source image.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mlib_ImageAdd(3MLIB)`, `mlib_ImageAdd_Fp(3MLIB)`, `mlib_ImageAdd_Inp(3MLIB)`, `attributes(5)`

mllib_ImageAdd_Inp(3MLIB)

NAME	mllib_ImageAdd_Inp – computes the addition of two images on a pixel-by-pixel basis, in place						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageAdd_Inp(mllib_image *<i>src1dst</i>, const mllib_image *<i>src2</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageAdd_Inp()</code> function computes the addition of two images on a pixel-by-pixel basis, in place.</p> <p>It uses the following equation:</p> $\text{src1dst}[x][y][i] = \text{src1dst}[x][y][i] + \text{src2}[x][y][i]$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>src1dst</i> Pointer to first source and destination image.</p> <p><i>src2</i> Pointer to second source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageAdd(3MLIB)</code> , <code>mllib_ImageAdd_Fp(3MLIB)</code> , <code>mllib_ImageAdd_Fp_Inp(3MLIB)</code> , <code>attributes(5)</code>						

NAME	mllib_ImageAffine – image affine transformation
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageAffine(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, const mllib_d64 *<i>mtx</i>, mllib_filter <i>filter</i>, mllib_edge <i>edge</i>);</pre>
DESCRIPTION	<p>The <code>mllib_ImageAffine()</code> function does affine transformation on an image according to the following equation:</p> $\begin{aligned} x_d &= a*x_s + b*y_s + t_x \\ y_d &= c*x_s + d*y_s + t_y \end{aligned}$ <p>where a point with coordinates (x_s, y_s) in the source image is mapped to a point with coordinates (x_d, y_d) in the destination image.</p> <p>The data type of the images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p> <p>The width and height of the destination image can be different from the width and height of the source image.</p> <p>The center of the upper-left corner pixel of an image is located at (0.5, 0.5).</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>mtx</i> Transformation matrix. <code>mtx[0]</code> holds a; <code>mtx[1]</code> holds b; <code>mtx[2]</code> holds t_x; <code>mtx[3]</code> holds c; <code>mtx[4]</code> holds d; <code>mtx[5]</code> holds t_y.</p> <p><i>filter</i> Type of resampling filter. It can be one of the following:</p> <pre>MLIB_NEAREST MLIB_BILINEAR MLIB_BICUBIC MLIB_BICUBIC2</pre> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_OP_NEAREST MLIB_EDGE_SRC_EXTEND MLIB_EDGE_SRC_PADDED</pre>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .

mllib_ImageAffine(3MLIB)

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageAffine_Fp(3MLIB)`, `mllib_ImageAffineIndex(3MLIB)`,
`mllib_ImageAffineTransform(3MLIB)`,
`mllib_ImageAffineTransform_Fp(3MLIB)`,
`mllib_ImageAffineTransformIndex(3MLIB)`,
`mllib_ImageSetPaddings(3MLIB)`, `attributes(5)`

NAME	mllib_ImageAffine_Fp – image affine transformation
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageAffine_Fp(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, const mllib_d64 *<i>mtx</i>, mllib_filter <i>filter</i>, mllib_edge <i>edge</i>);</pre>
DESCRIPTION	<p>The <code>mllib_ImageAffine_Fp()</code> function does affine transformation on a floating-point image according to the following equation:</p> $\begin{aligned} x_d &= a*x_s + b*y_s + t_x \\ y_d &= c*x_s + d*y_s + t_y \end{aligned}$ <p>where a point with coordinates (x_s, y_s) in the source image is mapped to a point with coordinates (x_d, y_d) in the destination image.</p> <p>The data type of the images can be <code>MLIB_FLOAT</code> or <code>MLIB_DOUBLE</code>.</p> <p>The width and height of the destination image can be different from the width and height of the source image.</p> <p>The center of the upper-left corner pixel of an image is located at (0.5, 0.5).</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>mtx</i> Transformation matrix. <code>mtx[0]</code> holds a; <code>mtx[1]</code> holds b; <code>mtx[2]</code> holds t_x; <code>mtx[3]</code> holds c; <code>mtx[4]</code> holds d; <code>mtx[5]</code> holds t_y.</p> <p><i>filter</i> Type of resampling filter. It can be one of the following:</p> <pre>MLIB_NEAREST MLIB_BILINEAR MLIB_BICUBIC MLIB_BICUBIC2</pre> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_OP_NEAREST MLIB_EDGE_SRC_EXTEND MLIB_EDGE_SRC_PADDED</pre>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:

mllib_ImageAffine_Fp(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

`mllib_ImageAffine(3MLIB)`, `mllib_ImageAffineIndex(3MLIB)`,
`mllib_ImageAffineTransform(3MLIB)`,
`mllib_ImageAffineTransform_Fp(3MLIB)`,
`mllib_ImageAffineTransformIndex(3MLIB)`,
`mllib_ImageSetPaddings(3MLIB)`, `attributes(5)`

NAME	mllib_ImageAffineIndex – affine transformation on a color indexed image
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageAffineIndex(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, const mllib_d64 *<i>mtx</i>, mllib_filter <i>filter</i>, mllib_edge <i>edge</i>, const void *<i>colormap</i>);</pre>
DESCRIPTION	<p>The <code>mllib_ImageAffineIndex()</code> function does affine transformation on a color indexed image according to the following equation:</p> $\begin{aligned} x_d &= a*x_s + b*y_s + t_x \\ y_d &= c*x_s + d*y_s + t_y \end{aligned}$ <p>where a point with coordinates (x_s, y_s) in the source image is mapped to a point with coordinates (x_d, y_d) in the destination image.</p> <p>The image data type must be <code>MLIB_BYTE</code> or <code>MLIB_SHORT</code>.</p> <p>The width and height of the destination image can be different from the width and height of the source image.</p> <p>The center of the upper-left corner pixel of an image is located at (0.5, 0.5).</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>mtx</i> Transformation matrix. <code>mtx[0]</code> holds a; <code>mtx[1]</code> holds b; <code>mtx[2]</code> holds t_x; <code>mtx[3]</code> holds c; <code>mtx[4]</code> holds d; <code>mtx[5]</code> holds t_y.</p> <p><i>filter</i> Type of resampling filter. It can be one of the following:</p> <pre>MLIB_NEAREST MLIB_BILINEAR MLIB_BICUBIC MLIB_BICUBIC2</pre> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_OP_NEAREST MLIB_EDGE_SRC_EXTEND MLIB_EDGE_SRC_PADDED</pre> <p><i>colormap</i> Internal data structure for inverse color mapping. This data structure is generated by the <code>mllib_ImageColorTrue2IndexInit()</code> function.</p>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .

mllib_ImageAffineIndex(3MLIB)

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageAffine(3MLIB)`, `mllib_ImageAffine_Fp(3MLIB)`,
`mllib_ImageAffineTransform(3MLIB)`,
`mllib_ImageAffineTransform_Fp(3MLIB)`,
`mllib_ImageAffineTransformIndex(3MLIB)`, `attributes(5)`

NAME	mllib_ImageAffineTable – affine transformation on an image with table-driven interpolation										
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageAffineTable(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, const mllib_d64 *<i>mtx</i>, const void *<i>interp_table</i>, mllib_edge <i>edge</i>);</pre>										
DESCRIPTION	<p>The <code>mllib_ImageAffineTable()</code> function does affine transformation on an image with table-driven interpolation.</p> <p>The following equation represents the affine transformation:</p> $\begin{aligned} x_d &= a*x_s + b*y_s + t_x \\ y_d &= c*x_s + d*y_s + t_y \end{aligned}$ <p>where a point with coordinates (x_s, y_s) in the source image is mapped to a point with coordinates (x_d, y_d) in the destination image.</p> <p>The data type of the images can be <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p> <p>The width and height of the destination image can be different from the width and height of the source image.</p> <p>The center of the upper-left corner pixel of an image is located at (0.5, 0.5).</p>										
PARAMETERS	<p>The function takes the following arguments:</p> <table border="0"> <tr> <td style="padding-right: 20px;"><i>dst</i></td> <td>Pointer to destination image.</td> </tr> <tr> <td><i>src</i></td> <td>Pointer to source image.</td> </tr> <tr> <td><i>mtx</i></td> <td>Transformation matrix. <code>mtx[0]</code> holds a; <code>mtx[1]</code> holds b; <code>mtx[2]</code> holds t_x; <code>mtx[3]</code> holds c; <code>mtx[4]</code> holds d; <code>mtx[5]</code> holds t_y.</td> </tr> <tr> <td><i>interp_table</i></td> <td>Pointer to an interpolation table. The table is created by the <code>mllib_ImageInterpTableCreate()</code> function.</td> </tr> <tr> <td><i>edge</i></td> <td>Type of edge condition. It can be one of the following:</td> </tr> </table> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_OP_NEAREST MLIB_EDGE_SRC_EXTEND MLIB_EDGE_SRC_PADDED</pre>	<i>dst</i>	Pointer to destination image.	<i>src</i>	Pointer to source image.	<i>mtx</i>	Transformation matrix. <code>mtx[0]</code> holds a ; <code>mtx[1]</code> holds b ; <code>mtx[2]</code> holds t_x ; <code>mtx[3]</code> holds c ; <code>mtx[4]</code> holds d ; <code>mtx[5]</code> holds t_y .	<i>interp_table</i>	Pointer to an interpolation table. The table is created by the <code>mllib_ImageInterpTableCreate()</code> function.	<i>edge</i>	Type of edge condition. It can be one of the following:
<i>dst</i>	Pointer to destination image.										
<i>src</i>	Pointer to source image.										
<i>mtx</i>	Transformation matrix. <code>mtx[0]</code> holds a ; <code>mtx[1]</code> holds b ; <code>mtx[2]</code> holds t_x ; <code>mtx[3]</code> holds c ; <code>mtx[4]</code> holds d ; <code>mtx[5]</code> holds t_y .										
<i>interp_table</i>	Pointer to an interpolation table. The table is created by the <code>mllib_ImageInterpTableCreate()</code> function.										
<i>edge</i>	Type of edge condition. It can be one of the following:										
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .										

mllib_ImageAffineTable(3MLIB)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageInterpTableCreate(3MLIB)`,
`mllib_ImageInterpTableDelete(3MLIB)`,
`mllib_ImageAffineTable_Fp(3MLIB)`, `mllib_ImageAffine(3MLIB)`,
`mllib_ImageAffine_Fp(3MLIB)`, `attributes(5)`

NAME	mllib_ImageAffineTable_Fp – affine transformation on an image with table-driven interpolation										
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageAffineTable_Fp(mllib_image *dst, const mllib_image *src, const mllib_d64 *mtx, const void *interp_table, mllib_edge edge);</pre>										
DESCRIPTION	<p>The <code>mllib_ImageAffineTable_Fp()</code> function does affine transformation on a floating-point image with table-driven interpolation.</p> <p>The following equation represents the affine transformation:</p> $\begin{aligned} x_d &= a*x_s + b*y_s + t_x \\ y_d &= c*x_s + d*y_s + t_y \end{aligned}$ <p>where a point with coordinates (x_s, y_s) in the source image is mapped to a point with coordinates (x_d, y_d) in the destination image.</p> <p>The data type of the images can be <code>MLIB_FLOAT</code> or <code>MLIB_DOUBLE</code>.</p> <p>The width and height of the destination image can be different from the width and height of the source image.</p> <p>The center of the upper-left corner pixel of an image is located at $(0.5, 0.5)$.</p>										
PARAMETERS	<p>The function takes the following arguments:</p> <table border="0"> <tr> <td style="padding-right: 20px;"><i>dst</i></td> <td>Pointer to destination image.</td> </tr> <tr> <td><i>src</i></td> <td>Pointer to source image.</td> </tr> <tr> <td><i>mtx</i></td> <td>Transformation matrix. <code>mtx[0]</code> holds <i>a</i>; <code>mtx[1]</code> holds <i>b</i>; <code>mtx[2]</code> holds <i>t_x</i>; <code>mtx[3]</code> holds <i>c</i>; <code>mtx[4]</code> holds <i>d</i>; <code>mtx[5]</code> holds <i>t_y</i>.</td> </tr> <tr> <td><i>interp_table</i></td> <td>Pointer to an interpolation table. The table is created by the <code>mllib_ImageInterpTableCreate()</code> function.</td> </tr> <tr> <td><i>edge</i></td> <td>Type of edge condition. It can be one of the following:</td> </tr> </table> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_OP_NEAREST MLIB_EDGE_SRC_EXTEND MLIB_EDGE_SRC_PADDED</pre>	<i>dst</i>	Pointer to destination image.	<i>src</i>	Pointer to source image.	<i>mtx</i>	Transformation matrix. <code>mtx[0]</code> holds <i>a</i> ; <code>mtx[1]</code> holds <i>b</i> ; <code>mtx[2]</code> holds <i>t_x</i> ; <code>mtx[3]</code> holds <i>c</i> ; <code>mtx[4]</code> holds <i>d</i> ; <code>mtx[5]</code> holds <i>t_y</i> .	<i>interp_table</i>	Pointer to an interpolation table. The table is created by the <code>mllib_ImageInterpTableCreate()</code> function.	<i>edge</i>	Type of edge condition. It can be one of the following:
<i>dst</i>	Pointer to destination image.										
<i>src</i>	Pointer to source image.										
<i>mtx</i>	Transformation matrix. <code>mtx[0]</code> holds <i>a</i> ; <code>mtx[1]</code> holds <i>b</i> ; <code>mtx[2]</code> holds <i>t_x</i> ; <code>mtx[3]</code> holds <i>c</i> ; <code>mtx[4]</code> holds <i>d</i> ; <code>mtx[5]</code> holds <i>t_y</i> .										
<i>interp_table</i>	Pointer to an interpolation table. The table is created by the <code>mllib_ImageInterpTableCreate()</code> function.										
<i>edge</i>	Type of edge condition. It can be one of the following:										
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .										

mllib_ImageAffineTable_Fp(3MLIB)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageInterpTableCreate(3MLIB)`,
`mllib_ImageInterpTableDelete(3MLIB)`, `mllib_ImageAffineTable(3MLIB)`,
`mllib_ImageAffine(3MLIB)`, `mllib_ImageAffine_Fp(3MLIB)`, `attributes(5)`

NAME	mllib_ImageAffineTransform – affine transformation on an image, checking the matrix first
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageAffineTransform(mllib_image *dst, const mllib_image *src, const mllib_d64 *mtx, mllib_filter filter, mllib_edge edge);</pre>
DESCRIPTION	<p>The <code>mllib_ImageAffineTransform()</code> function does affine transformation on an image, checking the matrix first and taking advantage of special cases.</p> <p>The following equation represents the affine transformation:</p> $\begin{aligned} x_d &= a*x_s + b*y_s + t_x \\ y_d &= c*x_s + d*y_s + t_y \end{aligned}$ <p>where a point with coordinates (x_s, y_s) in the source image is mapped to a point with coordinates (x_d, y_d) in the destination image.</p> <p>The data type of the images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p> <p>The width and height of the destination image can be different from the width and height of the source image.</p> <p>The center of the upper-left corner pixel of an image is located at $(0.5, 0.5)$.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>mtx</i> Transformation matrix. <code>mtx[0]</code> holds <i>a</i>; <code>mtx[1]</code> holds <i>b</i>; <code>mtx[2]</code> holds <i>t_x</i>; <code>mtx[3]</code> holds <i>c</i>; <code>mtx[4]</code> holds <i>d</i>; <code>mtx[5]</code> holds <i>t_y</i>.</p> <p><i>filter</i> Type of resampling filter. It can be one of the following:</p> <p style="margin-left: 40px;"><code>MLIB_NEAREST</code> <code>MLIB_BILINEAR</code> <code>MLIB_BICUBIC</code> <code>MLIB_BICUBIC2</code></p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <p style="margin-left: 40px;"><code>MLIB_EDGE_DST_NO_WRITE</code> <code>MLIB_EDGE_DST_FILL_ZERO</code> <code>MLIB_EDGE_OP_NEAREST</code> <code>MLIB_EDGE_SRC_EXTEND</code> <code>MLIB_EDGE_SRC_PADDED</code></p>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .

mllib_ImageAffineTransform(3MLIB)

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageAffine(3MLIB)`, `mllib_ImageAffine_Fp(3MLIB)`,
`mllib_ImageAffineIndex(3MLIB)`, `mllib_ImageAffineTransform_Fp(3MLIB)`,
`mllib_ImageAffineTransformIndex(3MLIB)`,
`mllib_ImageSetPaddings(3MLIB)`, `attributes(5)`

NAME	mllib_ImageAffineTransform_Fp – affine transformation on an image, checking the matrix first
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageAffineTransform_Fp(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, const mllib_d64 *<i>mtx</i>, mllib_filter <i>filter</i>, mllib_edge <i>edge</i>);</pre>
DESCRIPTION	<p>The <code>mllib_ImageAffineTransform_Fp()</code> function does affine transformation on a floating-point image, checking the matrix first and taking advantage of special cases.</p> <p>The following equation represents the affine transformation:</p> $\begin{aligned} x_d &= a*x_s + b*y_s + t_x \\ y_d &= c*x_s + d*y_s + t_y \end{aligned}$ <p>where a point with coordinates (x_s, y_s) in the source image is mapped to a point with coordinates (x_d, y_d) in the destination image.</p> <p>The data type of the images can be <code>MLIB_FLOAT</code> or <code>MLIB_DOUBLE</code>.</p> <p>The width and height of the destination image can be different from the width and height of the source image.</p> <p>The center of the upper-left corner pixel of an image is located at (0.5, 0.5).</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>mtx</i> Transformation matrix. <code>mtx[0]</code> holds a; <code>mtx[1]</code> holds b; <code>mtx[2]</code> holds t_x; <code>mtx[3]</code> holds c; <code>mtx[4]</code> holds d; <code>mtx[5]</code> holds t_y.</p> <p><i>filter</i> Type of resampling filter. It can be one of the following:</p> <p style="margin-left: 40px;"><code>MLIB_NEAREST</code> <code>MLIB_BILINEAR</code> <code>MLIB_BICUBIC</code> <code>MLIB_BICUBIC2</code></p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <p style="margin-left: 40px;"><code>MLIB_EDGE_DST_NO_WRITE</code> <code>MLIB_EDGE_DST_FILL_ZERO</code> <code>MLIB_EDGE_OP_NEAREST</code> <code>MLIB_EDGE_SRC_EXTEND</code> <code>MLIB_EDGE_SRC_PADDED</code></p>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .

mllib_ImageAffineTransform_Fp(3MLIB)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageAffine(3MLIB)`, `mllib_ImageAffine_Fp(3MLIB)`,
`mllib_ImageAffineIndex(3MLIB)`, `mllib_ImageAffineTransform(3MLIB)`,
`mllib_ImageAffineTransformIndex(3MLIB)`,
`mllib_ImageSetPaddings(3MLIB)`, `attributes(5)`

mllib_ImageAffineTransformIndex(3MLIB)

NAME	mllib_ImageAffineTransformIndex – affine transformation on a color indexed image, checking the matrix first												
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageAffineTransformIndex(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, const mllib_d64 *<i>mtx</i>, mllib_filter <i>filter</i>, mllib_edge <i>edge</i>, const void *<i>colormap</i>);</pre>												
DESCRIPTION	<p>The <code>mllib_ImageAffineTransformIndex()</code> function does affine transformation on a color indexed image, checking the matrix first and taking advantage of special cases.</p> <p>The following equation represents the affine transformation:</p> $\begin{aligned}x_d &= a*x_s + b*y_s + t_x \\y_d &= c*x_s + d*y_s + t_y\end{aligned}$ <p>where a point with coordinates (x_s, y_s) in the source image is mapped to a point with coordinates (x_d, y_d) in the destination image.</p> <p>The image data type must be <code>MLIB_BYTE</code> or <code>MLIB_SHORT</code>.</p> <p>The width and height of the destination image can be different from the width and height of the source image.</p> <p>The center of the upper-left corner pixel of an image is located at (0.5, 0.5).</p>												
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>dst</i></td><td>Pointer to destination image.</td></tr><tr><td><i>src</i></td><td>Pointer to source image.</td></tr><tr><td><i>mtx</i></td><td>Transformation matrix. <code>mtx[0]</code> holds a; <code>mtx[1]</code> holds b; <code>mtx[2]</code> holds t_x; <code>mtx[3]</code> holds c; <code>mtx[4]</code> holds d; <code>mtx[5]</code> holds t_y.</td></tr><tr><td><i>filter</i></td><td>Type of resampling filter. It can be one of the following: <code>MLIB_NEAREST</code> <code>MLIB_BILINEAR</code> <code>MLIB_BICUBIC</code> <code>MLIB_BICUBIC2</code></td></tr><tr><td><i>edge</i></td><td>Type of edge condition. It can be one of the following: <code>MLIB_EDGE_DST_NO_WRITE</code> <code>MLIB_EDGE_DST_FILL_ZERO</code> <code>MLIB_EDGE_OP_NEAREST</code> <code>MLIB_EDGE_SRC_EXTEND</code> <code>MLIB_EDGE_SRC_PADDED</code></td></tr><tr><td><i>colormap</i></td><td>Internal data structure for inverse color mapping. This data structure is generated by the <code>mllib_ImageColorTrue2IndexInit()</code> function.</td></tr></table>	<i>dst</i>	Pointer to destination image.	<i>src</i>	Pointer to source image.	<i>mtx</i>	Transformation matrix. <code>mtx[0]</code> holds a ; <code>mtx[1]</code> holds b ; <code>mtx[2]</code> holds t_x ; <code>mtx[3]</code> holds c ; <code>mtx[4]</code> holds d ; <code>mtx[5]</code> holds t_y .	<i>filter</i>	Type of resampling filter. It can be one of the following: <code>MLIB_NEAREST</code> <code>MLIB_BILINEAR</code> <code>MLIB_BICUBIC</code> <code>MLIB_BICUBIC2</code>	<i>edge</i>	Type of edge condition. It can be one of the following: <code>MLIB_EDGE_DST_NO_WRITE</code> <code>MLIB_EDGE_DST_FILL_ZERO</code> <code>MLIB_EDGE_OP_NEAREST</code> <code>MLIB_EDGE_SRC_EXTEND</code> <code>MLIB_EDGE_SRC_PADDED</code>	<i>colormap</i>	Internal data structure for inverse color mapping. This data structure is generated by the <code>mllib_ImageColorTrue2IndexInit()</code> function.
<i>dst</i>	Pointer to destination image.												
<i>src</i>	Pointer to source image.												
<i>mtx</i>	Transformation matrix. <code>mtx[0]</code> holds a ; <code>mtx[1]</code> holds b ; <code>mtx[2]</code> holds t_x ; <code>mtx[3]</code> holds c ; <code>mtx[4]</code> holds d ; <code>mtx[5]</code> holds t_y .												
<i>filter</i>	Type of resampling filter. It can be one of the following: <code>MLIB_NEAREST</code> <code>MLIB_BILINEAR</code> <code>MLIB_BICUBIC</code> <code>MLIB_BICUBIC2</code>												
<i>edge</i>	Type of edge condition. It can be one of the following: <code>MLIB_EDGE_DST_NO_WRITE</code> <code>MLIB_EDGE_DST_FILL_ZERO</code> <code>MLIB_EDGE_OP_NEAREST</code> <code>MLIB_EDGE_SRC_EXTEND</code> <code>MLIB_EDGE_SRC_PADDED</code>												
<i>colormap</i>	Internal data structure for inverse color mapping. This data structure is generated by the <code>mllib_ImageColorTrue2IndexInit()</code> function.												

`mllib_ImageAffineTransformIndex(3MLIB)`

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageAffine(3MLIB)`, `mllib_ImageAffine_Fp(3MLIB)`, `mllib_ImageAffineIndex(3MLIB)`, `mllib_ImageAffineTransform(3MLIB)`, `mllib_ImageAffineTransform_Fp(3MLIB)`, `attributes(5)`

NAME	mlib_ImageAnd – computes the And of two images						
SYNOPSIS	<pre>cc [flag...] file... -lmlib [library...] #include <mlib.h> mlib_status mlib_ImageAnd(mlib_image *dst, const mlib_image *src1, const mlib_image *src2);</pre>						
DESCRIPTION	<p>The <code>mlib_ImageAnd()</code> function computes the And of two images according to the following equation:</p> $dst[x][y][i] = src1[x][y][i] \& src2[x][y][i]$ <p>The data type of the images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src1</i> Pointer to first source image.</p> <p><i>src2</i> Pointer to second source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mlib_ImageAnd_Inp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageAnd_Inp(3MLIB)

NAME | mllib_ImageAnd_Inp – computes the And of two image, in place

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageAnd_Inp(mllib_image *src1dst, const mllib_image
    *src2) ;
```

DESCRIPTION | The `mllib_ImageAnd_Inp()` function computes the And of two images, in place, according to the following equation:

$$\text{src1dst}[x][y][i] = \text{src1dst}[x][y][i] \ \& \ \text{src2}[x][y][i]$$

The data type of the images can be `MLIB_BIT`, `MLIB_BYTE`, `MLIB_SHORT`, `MLIB_USHORT`, or `MLIB_INT`.

PARAMETERS | The function takes the following arguments:

src1dst | Pointer to first source and destination image.

src2 | Pointer to second source image.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_ImageAnd\(3MLIB\)](#), [attributes\(5\)](#)

NAME mllib_ImageAndNot1_Inp – computes the And of the first source image and the Not of the second source image, in place

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageAndNot1_Inp(mllib_image *src1dst, const
    mllib_image *src2);
```

DESCRIPTION The mllib_ImageAndNot1_Inp() function computes the logical Not of the second source image and then computes the logical And of that result with the first source image, on a pixel-by-pixel basis, and stores the final result in the first source image. It uses the following equation:

$$src1dst[x][y][i] = src1dst[x][y][i] \& (\sim src2[x][y][i])$$

The data type of the images can be MLIB_BIT, MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, or MLIB_INT.

PARAMETERS The function takes the following arguments:

src1dst Pointer to first source and destination image.

src2 Pointer to second source image.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageAndNot(3MLIB), mllib_ImageAndNot2_Inp(3MLIB), attributes(5)

mllib_ImageAndNot2_Inp(3MLIB)

NAME | mllib_ImageAndNot2_Inp – computes the And of the first source image and the Not of the second source image, in place

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
mllib_status mllib_ImageAndNot2_Inp(mllib_image *src2dst, const  
mllib_image *src1);
```

DESCRIPTION | The mllib_ImageAndNot2_Inp() function computes the logical Not of the second source image and then computes the logical And of that result with the first source image, on a pixel-by-pixel basis, and stores the final result in the second source image. It uses the following equation:
$$\text{src2dst}[x][y][i] = \text{src1}[x][y][i] \ \& \ (\sim\text{src2dst}[x][y][i])$$

The data type of the images can be MLLIB_BIT, MLLIB_BYTE, MLLIB_SHORT, MLLIB_USHORT, or MLLIB_INT.

PARAMETERS | The function takes the following arguments:
src2dst Pointer to second source and destination image.
src1 Pointer to first source image.

RETURN VALUES | The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_ImageAnd\(3MLIB\)](#), [mllib_ImageAnd_Inp\(3MLIB\)](#), [attributes\(5\)](#)

NAME mllib_ImageAndNot – computes the And of the first source image and the Not of the second source image

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageAndNot(mllib_image *dst, const mllib_image
    *src1, const mllib_image *src2);
```

DESCRIPTION The mllib_ImageAndNot () function computes the logical Not of the second source image and then computes the logical And of the result with the first source image, on a pixel-by-pixel basis. It uses the following equation:

$$dst[x][y][i] = src1[x][y][i] \& (\sim src2[x][y][i])$$

The data type of the images can be MLIB_BIT, MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, or MLIB_INT.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src1 Pointer to first source image.

src2 Pointer to second source image.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageAndNot1_Inp(3MLIB), mllib_ImageAndNot2_Inp(3MLIB), attributes(5)

mllib_ImageAutoCorrel(3MLIB)

NAME	mllib_ImageAutoCorrel – auto-correlation of an image						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageAutoCorrel(mllib_d64 *correl, const mllib_image *img, mllib_s32 dx, mllib_s32 dy);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageAutoCorrel()</code> function computes the auto-correlation of an image, given an offset.</p> <p>It uses the following equation:</p> $\text{correl}[i] = \frac{1}{(w-dx) * (h-dy)} * \sum_{x=0}^{w-dx-1} \sum_{y=0}^{h-dy-1} (\text{img}[x][y][i] * \text{img}[x+dx][y+dy][i])$ <p>where <code>w</code> and <code>h</code> are the width and height of the image, respectively.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>correl</i> Pointer to auto-correlation array where size is equal to the number of channels. <code>correl[i]</code> contains the auto-correlation of channel <code>i</code>.</p> <p><i>img</i> Pointer to image.</p> <p><i>dx</i> Displacement in the X direction.</p> <p><i>dy</i> Displacement in the Y direction.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageAutoCorrel_Fp(3MLIB)</code> , <code>mllib_ImageCrossCorrel(3MLIB)</code> , <code>mllib_ImageCrossCorrel_Fp(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageAutoCorrel_Fp – auto-correlation of an image

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageAutoCorrel_Fp(mllib_d64 *correl, const
    mllib_image *img, mllib_s32 dx, mllib_s32 dy);
```

DESCRIPTION The mllib_ImageAutoCorrel_Fp() function computes the auto-correlation of a floating-point image, given an offset.

It uses the following equation:

$$\text{correl}[i] = \frac{1}{(w-dx) * (h-dy)} * \sum_{x=0}^{w-dx-1} \sum_{y=0}^{h-dy-1} (\text{img}[x][y][i] * \text{img}[x+dx][y+dy][i])$$

where w and h are the width and height of the image, respectively.

PARAMETERS The function takes the following arguments:

- correl* Pointer to auto-correlation array where size is equal to the number of channels. *correl*[i] contains the auto-correlation of channel i.
- img* Pointer to image.
- dx* Displacement in the X direction.
- dy* Displacement in the Y direction.

RETURN VALUES The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageAutoCorrel(3MLIB), mllib_ImageCrossCorrel(3MLIB), mllib_ImageCrossCorrel_Fp(3MLIB), attributes(5)

mllib_ImageAve(3MLIB)

NAME | mllib_ImageAve – average of two images

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageAve(mllib_image *dst, const mllib_image *src1,
    const mllib_image *src2);
```

DESCRIPTION | The `mllib_ImageAve()` function computes the average of two images on a pixel-by-pixel basis.

It uses the following equation:

$$dst[x][y][i] = (src1[x][y][i] + src2[x][y][i]) / 2$$

PARAMETERS | The function takes the following arguments:

dst | Pointer to destination image.

src1 | Pointer to first source image.

src2 | Pointer to second source image.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageAve_Fp(3MLIB)`, `mllib_ImageAve_Fp_Inp(3MLIB)`, `mllib_ImageAve_Inp(3MLIB)`, `attributes(5)`

- NAME** mllib_ImageAve_Fp – average of two images
- SYNOPSIS**

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageAve_Fp(mllib_image *dst, const mllib_image
    *src1, const mllib_image *src2);
```
- DESCRIPTION** The `mllib_ImageAve_Fp()` function computes the average of two floating-point images on a pixel-by-pixel basis.
- It uses the following equation:
- $$dst[x][y][i] = (src1[x][y][i] + src2[x][y][i]) / 2$$
- PARAMETERS** The function takes the following arguments:
- dst* Pointer to destination image.
- src1* Pointer to first source image.
- src2* Pointer to second source image.
- RETURN VALUES** The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.
- ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

- SEE ALSO** `mllib_ImageAve(3MLIB)`, `mllib_ImageAve_Fp_Inp(3MLIB)`, `mllib_ImageAve_Inp(3MLIB)`, `attributes(5)`

mllib_ImageAve_Fp_Inp(3MLIB)

NAME | mllib_ImageAve_Fp_Inp – average of two images, in place

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageAve_Fp_Inp(mllib_image *src1dst, const
mllib_image *src2);
```

DESCRIPTION | The `mllib_ImageAve_Fp_Inp()` function computes the average of two floating-point images on a pixel-by-pixel basis, in place.

It uses the following equation:

$$\text{src1dst}[x][y][i] = (\text{src1dst}[x][y][i] + \text{src2}[x][y][i]) / 2$$

PARAMETERS | The function takes the following arguments:

src1dst Pointer to first source and destination image.

src2 Pointer to second source image.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageAve(3MLIB)`, `mllib_ImageAve_Fp(3MLIB)`, `mllib_ImageAve_Inp(3MLIB)`, `attributes(5)`

NAME	mllib_ImageAve_Inp – average of two images, in place						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageAve_Inp(mllib_image *<i>src1dst</i>, const mllib_image *<i>src2</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageAve_Inp()</code> function computes the average of two images on a pixel-by-pixel basis, in place.</p> <p>It uses the following equation:</p> $\text{src1dst}[x][y][i] = (\text{src1dst}[x][y][i] + \text{src2}[x][y][i]) / 2$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>src1dst</i> Pointer to first source and destination image.</p> <p><i>src2</i> Pointer to second source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageAve(3MLIB)</code> , <code>mllib_ImageAve_Fp(3MLIB)</code> , <code>mllib_ImageAve_Fp_Inp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageBlend1_Fp_Inp(3MLIB)

NAME	mllib_ImageBlend1_Fp_Inp – blend with an alpha image						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageBlend1_Fp_Inp(mllib_image *src1dst, const mllib_image *src2, const mllib_image *alpha);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageBlend1_Fp_Inp()</code> function blends two images together, in place, on a pixel-by-pixel basis using an alpha image, when alpha is also on a pixel basis. The <i>alpha</i> image can be a single-channel image or have the same number of channels as the source and destination images.</p> <p>It uses the following equation when the <i>alpha</i> image is a single-channel image:</p> $\text{src1dst}[x][y][i] = \text{alpha}[x][y][0] * \text{src1dst}[x][y][i] + (1 - \text{alpha}[x][y][0]) * \text{src2}[x][y][i]$ <p>It uses the following equation when the <i>alpha</i> image has the same number of channels as the source and destination images:</p> $\text{src1dst}[x][y][i] = \text{alpha}[x][y][i] * \text{src1dst}[x][y][i] + (1 - \text{alpha}[x][y][i]) * \text{src2}[x][y][i]$						
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>src1dst</i></td><td>Pointer to first source and destination image.</td></tr><tr><td><i>src2</i></td><td>Pointer to second source image.</td></tr><tr><td><i>alpha</i></td><td>Alpha image used to control blending. The pixels in this image should have values in the range of [0.0,1.0].</td></tr></table>	<i>src1dst</i>	Pointer to first source and destination image.	<i>src2</i>	Pointer to second source image.	<i>alpha</i>	Alpha image used to control blending. The pixels in this image should have values in the range of [0.0,1.0].
<i>src1dst</i>	Pointer to first source and destination image.						
<i>src2</i>	Pointer to second source image.						
<i>alpha</i>	Alpha image used to control blending. The pixels in this image should have values in the range of [0.0,1.0].						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageBlend(3MLIB)</code> , <code>mllib_ImageBlend_Fp(3MLIB)</code> , <code>mllib_ImageBlend1_Inp(3MLIB)</code> , <code>mllib_ImageBlend2_Fp_Inp(3MLIB)</code> , <code>mllib_ImageBlend2_Inp(3MLIB)</code> , <code>attributes(5)</code>						

NAME	mllib_ImageBlend1_Inp – blend with an alpha image, in place						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageBlend1_Inp(mllib_image *src1dst, const mllib_image *src2, const mllib_image *alpha);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageBlend1_Inp()</code> function blends two images together, in place, on a pixel-by-pixel basis using an alpha image, when alpha is also on a pixel basis. The <i>alpha</i> image can be a single-channel image or have the same number of channels as the source and destination images.</p> <p>It uses the following equation when the <i>alpha</i> image is a single-channel image:</p> $\text{src1dst}[x][y][i] = a[x][y][0] * \text{src1dst}[x][y][i] + (1 - a[x][y][0]) * \text{src2}[x][y][i]$ <p>It uses the following equation when the <i>alpha</i> image has the same number of channels as the source and destination images:</p> $\text{src1dst}[x][y][i] = a[x][y][i] * \text{src1dst}[x][y][i] + (1 - a[x][y][i]) * \text{src2}[x][y][i]$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>src1dst</i> Pointer to first source and destination image.</p> <p><i>src2</i> Pointer to second source image.</p> <p><i>alpha</i> Alpha image used to control blending. The <i>a</i> value equals ($\text{alpha} * 2^{**(-8)}$) for <code>MLIB_BYTE</code> image, ($\text{alpha} * 2^{**(-15)}$) for <code>MLIB_SHORT</code> image, ($\text{alpha} * 2^{**(-16)}$) for <code>MLIB_USHORT</code> image, and ($\text{alpha} * 2^{**(-31)}$) for <code>MLIB_INT</code> image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageBlend(3MLIB)</code> , <code>mllib_ImageBlend_Fp(3MLIB)</code> , <code>mllib_ImageBlend1_Fp_Inp(3MLIB)</code> , <code>mllib_ImageBlend2_Fp_Inp(3MLIB)</code> , <code>mllib_ImageBlend2_Inp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageBlend2_Fp_Inp(3MLIB)

NAME	mllib_ImageBlend2_Fp_Inp – blend with an alpha image						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageBlend2_Fp_Inp(mllib_image *src2dst, const mllib_image *src1, const mllib_image *alpha);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageBlend2_Fp_Inp()</code> function blends two images together, in place, on a pixel-by-pixel basis using an alpha image, when a is also on a pixel basis. The <i>alpha</i> image can be a single-channel image or have the same number of channels as the source and destination images.</p> <p>It uses the following equation when the <i>alpha</i> image is a single-channel image:</p> $\text{src2dst}[x][y][i] = \text{alpha}[x][y][0] * \text{src1}[x][y][i] + (1 - \text{alpha}[x][y][0]) * \text{src2dst}[x][y][i]$ <p>It uses the following equation when the <i>alpha</i> image has the same number of channels as the source and destination images:</p> $\text{src2dst}[x][y][i] = \text{alpha}[x][y][i] * \text{src1}[x][y][i] + (1 - \text{alpha}[x][y][i]) * \text{src2dst}[x][y][i]$ <p>It uses the following equation:</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>src2dst</i></td><td>Pointer to second source and destination image.</td></tr><tr><td><i>src1</i></td><td>Pointer to first source image.</td></tr><tr><td><i>alpha</i></td><td>Alpha image used to control blending. The pixels in this image should have values in the range of [0.0, 1.0].</td></tr></table>	<i>src2dst</i>	Pointer to second source and destination image.	<i>src1</i>	Pointer to first source image.	<i>alpha</i>	Alpha image used to control blending. The pixels in this image should have values in the range of [0.0, 1.0].
<i>src2dst</i>	Pointer to second source and destination image.						
<i>src1</i>	Pointer to first source image.						
<i>alpha</i>	Alpha image used to control blending. The pixels in this image should have values in the range of [0.0, 1.0].						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageBlend(3MLIB)</code> , <code>mllib_ImageBlend_Fp(3MLIB)</code> , <code>mllib_ImageBlend1_Fp_Inp(3MLIB)</code> , <code>mllib_ImageBlend1_Inp(3MLIB)</code> , <code>mllib_ImageBlend2_Inp(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageBlend2_Inp – blend with an alpha image, in place

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageBlend2_Inp(mllib_image *src2dst, const
    mllib_image *src1, const mllib_image *alpha);
```

DESCRIPTION

The `mllib_ImageBlend2_Inp()` function blends two images together, in place, on a pixel-by-pixel basis using an alpha image, when alpha is also on a pixel basis. The *alpha* image can be a single-channel image or have the same number of channels as the source and destination images.

It uses the following equation when the *alpha* image is a single-channel image:

$$\text{src2dst}[x][y][i] = a[x][y][0] * \text{src1}[x][y][i] + (1 - a[x][y][0]) * \text{src2dst}[x][y][i]$$

It uses the following equation when the *alpha* image has the same number of channels as the source and destination images:

$$\text{src2dst}[x][y][i] = a[x][y][i] * \text{src1}[x][y][i] + (1 - a[x][y][i]) * \text{src2dst}[x][y][i]$$

PARAMETERS

The function takes the following arguments:

src2dst Pointer to second source and destination image.

src1 Pointer to first source image.

alpha Alpha image used to control blending. The *a* value equals ($\text{alpha} * 2^{**(-8)}$) for `MLIB_BYTE` image, ($\text{alpha} * 2^{**(-15)}$) for `MLIB_SHORT` image, ($\text{alpha} * 2^{**(-16)}$) for `MLIB_USHORT` image, and ($\text{alpha} * 2^{**(-31)}$) for `MLIB_INT` image.

RETURN VALUES

The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageBlend(3MLIB)`, `mllib_ImageBlend_Fp(3MLIB)`, `mllib_ImageBlend1_Fp_Inp(3MLIB)`, `mllib_ImageBlend1_Inp(3MLIB)`, `mllib_ImageBlend2_Fp_Inp(3MLIB)`, `attributes(5)`

mllib_ImageBlend(3MLIB)

NAME	mllib_ImageBlend – blend with an alpha image								
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageBlend(mllib_image *dst, const mllib_image *src1, const mllib_image *src2, const mllib_image *alpha);</pre>								
DESCRIPTION	<p>The <code>mllib_ImageBlend()</code> function blends two images together on a pixel-by-pixel basis using an alpha image, when alpha is also on a pixel basis. The <i>alpha</i> image can be a single-channel image or have the same number of channels as the source and destination images.</p> <p>It uses the following equation when the <i>alpha</i> image is a single-channel image:</p> $dst[x][y][i] = a[x][y][0]*src1[x][y][i] + (1 - a[x][y][0])*src2[x][y][i]$ <p>It uses the following equation when the <i>alpha</i> image has the same number of channels as the source and destination images:</p> $dst[x][y][i] = a[x][y][i]*src1[x][y][i] + (1 - a[x][y][i])*src2[x][y][i]$								
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>dst</i></td><td>Pointer to destination image.</td></tr><tr><td><i>src1</i></td><td>Pointer to first source image.</td></tr><tr><td><i>src2</i></td><td>Pointer to second source image.</td></tr><tr><td><i>alpha</i></td><td>Alpha image used to control blending. The <i>a</i> value equals (alpha * 2**(-8)) for <code>MLIB_BYTE</code> image, (alpha * 2**(-15)) for <code>MLIB_SHORT</code> image, (alpha * 2**(-16)) for <code>MLIB_USHORT</code> image, and (alpha * 2**(-31)) for <code>MLIB_INT</code> image.</td></tr></table>	<i>dst</i>	Pointer to destination image.	<i>src1</i>	Pointer to first source image.	<i>src2</i>	Pointer to second source image.	<i>alpha</i>	Alpha image used to control blending. The <i>a</i> value equals (alpha * 2**(-8)) for <code>MLIB_BYTE</code> image, (alpha * 2**(-15)) for <code>MLIB_SHORT</code> image, (alpha * 2**(-16)) for <code>MLIB_USHORT</code> image, and (alpha * 2**(-31)) for <code>MLIB_INT</code> image.
<i>dst</i>	Pointer to destination image.								
<i>src1</i>	Pointer to first source image.								
<i>src2</i>	Pointer to second source image.								
<i>alpha</i>	Alpha image used to control blending. The <i>a</i> value equals (alpha * 2**(-8)) for <code>MLIB_BYTE</code> image, (alpha * 2**(-15)) for <code>MLIB_SHORT</code> image, (alpha * 2**(-16)) for <code>MLIB_USHORT</code> image, and (alpha * 2**(-31)) for <code>MLIB_INT</code> image.								
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .								
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:								
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe		
ATTRIBUTE TYPE	ATTRIBUTE VALUE								
Interface Stability	Evolving								
MT-Level	MT-Safe								
SEE ALSO	<code>mllib_ImageBlend_Fp(3MLIB)</code> , <code>mllib_ImageBlend1_Fp_Inp(3MLIB)</code> , <code>mllib_ImageBlend1_Inp(3MLIB)</code> , <code>mllib_ImageBlend2_Fp_Inp(3MLIB)</code> , <code>mllib_ImageBlend2_Inp(3MLIB)</code> , <code>attributes(5)</code>								

mllib_ImageBlend_BSRC1_BSRC2(3MLIB)

NAME mllib_ImageBlend_BSRC1_BSRC2, mllib_ImageBlend_DA_DA,
 mllib_ImageBlend_DA_DC, mllib_ImageBlend_DA_OMDA,
 mllib_ImageBlend_DA_OMDC, mllib_ImageBlend_DA_OMSA,
 mllib_ImageBlend_DA_ONE, mllib_ImageBlend_DA_SA, mllib_ImageBlend_DA_SAS,
 mllib_ImageBlend_DA_ZERO, mllib_ImageBlend_OMDA_DA,
 mllib_ImageBlend_OMDA_DC, mllib_ImageBlend_OMDA_OMDA,
 mllib_ImageBlend_OMDA_OMDC, mllib_ImageBlend_OMDA_OMSA,
 mllib_ImageBlend_OMDA_ONE, mllib_ImageBlend_OMDA_SA,
 mllib_ImageBlend_OMDA_SAS, mllib_ImageBlend_OMDA_ZERO,
 mllib_ImageBlend_OMSA_DA, mllib_ImageBlend_OMSA_DC,
 mllib_ImageBlend_OMSA_OMDA, mllib_ImageBlend_OMSA_OMDC,
 mllib_ImageBlend_OMSA_OMSA, mllib_ImageBlend_OMSA_ONE,
 mllib_ImageBlend_OMSA_SA, mllib_ImageBlend_OMSA_SAS,
 mllib_ImageBlend_OMSA_ZERO, mllib_ImageBlend_OMSC_DA,
 mllib_ImageBlend_OMSC_DC, mllib_ImageBlend_OMSC_OMDA,
 mllib_ImageBlend_OMSC_OMDC, mllib_ImageBlend_OMSC_OMSA,
 mllib_ImageBlend_OMSC_ONE, mllib_ImageBlend_OMSC_SA,
 mllib_ImageBlend_OMSC_SAS, mllib_ImageBlend_OMSC_ZERO,
 mllib_ImageBlend_ONE_DA, mllib_ImageBlend_ONE_DC,
 mllib_ImageBlend_ONE_OMDA, mllib_ImageBlend_ONE_OMDC,
 mllib_ImageBlend_ONE_OMSA, mllib_ImageBlend_ONE_ONE,
 mllib_ImageBlend_ONE_SA, mllib_ImageBlend_ONE_SAS,
 mllib_ImageBlend_ONE_ZERO, mllib_ImageBlend_SA_DA, mllib_ImageBlend_SA_DC,
 mllib_ImageBlend_SA_OMDA, mllib_ImageBlend_SA_OMDC,
 mllib_ImageBlend_SA_OMSA, mllib_ImageBlend_SA_ONE, mllib_ImageBlend_SA_SA,
 mllib_ImageBlend_SA_SAS, mllib_ImageBlend_SA_ZERO, mllib_ImageBlend_SC_DA,
 mllib_ImageBlend_SC_DC, mllib_ImageBlend_SC_OMDA,
 mllib_ImageBlend_SC_OMDC, mllib_ImageBlend_SC_OMSA,
 mllib_ImageBlend_SC_ONE, mllib_ImageBlend_SC_SA, mllib_ImageBlend_SC_SAS,
 mllib_ImageBlend_SC_ZERO, mllib_ImageBlend_ZERO_DA,
 mllib_ImageBlend_ZERO_DC, mllib_ImageBlend_ZERO_OMDA,
 mllib_ImageBlend_ZERO_OMDC, mllib_ImageBlend_ZERO_OMSA,
 mllib_ImageBlend_ZERO_ONE, mllib_ImageBlend_ZERO_SA,
 mllib_ImageBlend_ZERO_SAS, mllib_ImageBlend_ZERO_ZERO – blending

SYNOPSIS `cc [flag...] file... -lmllib [library...]
 #include <mllib.h>`

```
mllib_status mllib_ImageBlend_BSRC1_BSRC2(mllib_image *dst, const
      mllib_image *src1, const mllib_image *src2, mllib_s32 cmask);
```

DESCRIPTION This group of functions supports digital image composition. They are low-level, non-in-place, blending functions.

The image type must be `MLIB_BYTE`. The input and output images must contain three or four channels. For three-channel images, the alpha value is as if the alpha value is 1.

BSRC1 is one of the following: ZERO, ONE, SC, OMSC, DA, SA, OMDA, or OMSA. BSRC2 is one of the following: ZERO, ONE, DC, OMDC, DA, SA, OMDA, OMSA, or SAS.

mllib_ImageBlend_BSRC1_BSRC2(3MLIB)

The following are predefined blend factor types used in mediaLib image composition functions.

```
/* image blend factors */
typedef enum {
    MLIB_BLEND_ZERO,
    MLIB_BLEND_ONE,
    MLIB_BLEND_DST_COLOR,
    MLIB_BLEND_SRC_COLOR,
    MLIB_BLEND_ONE_MINUS_DST_COLOR,
    MLIB_BLEND_ONE_MINUS_SRC_COLOR,
    MLIB_BLEND_DST_ALPHA,
    MLIB_BLEND_SRC_ALPHA,
    MLIB_BLEND_ONE_MINUS_DST_ALPHA,
    MLIB_BLEND_ONE_MINUS_SRC_ALPHA,
    MLIB_BLEND_SRC_ALPHA_SATURATE
} mlib_blend;
```

See the following table for the definitions of the blend factors.

Type	Blend Factor [*]	Abbr.
MLIB_BLEND_ZERO	(0,0,0,0)	ZERO
MLIB_BLEND_ONE	(1,1,1,1)	ONE
MLIB_BLEND_DST_COLOR	(Rd,Gd,Bd,Ad)	DC
MLIB_BLEND_SRC_COLOR	(Rs,Gs,Bs,As)	SC
MLIB_BLEND_ONE_MINUS_DST_COLOR	(1,1,1,1)-(Rd,Gd,Bd,Ad)	OMDC
MLIB_BLEND_ONE_MINUS_SRC_COLOR	(1,1,1,1)-(Rs,Gs,Bs,As)	OMSC
MLIB_BLEND_DST_ALPHA	(Ad,Ad,Ad,Ad)	DA
MLIB_BLEND_SRC_ALPHA	(As,As,As,As)	SA
MLIB_BLEND_ONE_MINUS_DST_ALPHA	(1,1,1,1)-(Ad,Ad,Ad,Ad)	OMDA
MLIB_BLEND_ONE_MINUS_SRC_ALPHA	(1,1,1,1)-(As,As,As,As)	OMSA
MLIB_BLEND_SRC_ALPHA_SATURATE	(f,f,f,1)	SAS

[*]: The components of the first source image pixel are (Rd,Gd,Bd,Ad), and the components of the second source pixel are (Rs,Gs,Bs,As). Function $f = \min(As, 1-Ad)$.

The blending formula for non-in-place processing is:

$$C_d = C_{s1} * S_1 + C_{s2} * S_2$$

where C_d is the destination pixel (Rd,Gd,Bd,Ad), C_{s1} is the first source pixel (Rs1,Gs1,Bs1,As1), C_{s2} is the second source pixel (Rs2,Gs2,Bs2,As2), and S_1 and S_2 are the blend factors for the first and second sources, respectively.

PARAMETERS Each of the functions takes the following arguments:

`mllib_ImageBlend_BSRC1_BSRC2(3MLIB)`

dst Pointer to destination image.
src1 Pointer to the first source image.
src2 Pointer to the second source image.
cmask Channel mask to indicate the alpha channel. Each bit of the mask represents a channel in the image. The channel corresponding to the 1 bit is the alpha channel. *cmask* must be either 0x01 or 0x08.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageBlend_BSRC1_BSRC2_Inp(3MLIB)`, `mllib_ImageComposite(3MLIB)`, `mllib_ImageComposite_Inp(3MLIB)`, `attributes(5)`

mllib_ImageBlend_BSRC1_BSRC2_Inp(3MLIB)

NAME	mllib_ImageBlend_BSRC1_BSRC2_Inp, mllib_ImageBlend_DA_DA_Inp, mllib_ImageBlend_DA_DC_Inp, mllib_ImageBlend_DA_OMDA_Inp, mllib_ImageBlend_DA_OMDC_Inp, mllib_ImageBlend_DA_OMSA_Inp, mllib_ImageBlend_DA_ONE_Inp, mllib_ImageBlend_DA_SA_Inp, mllib_ImageBlend_DA_SAS_Inp, mllib_ImageBlend_DA_ZERO_Inp, mllib_ImageBlend_OMDA_DA_Inp, mllib_ImageBlend_OMDA_DC_Inp, mllib_ImageBlend_OMDA_OMDA_Inp, mllib_ImageBlend_OMDA_OMDC_Inp, mllib_ImageBlend_OMDA_OMSA_Inp, mllib_ImageBlend_OMDA_ONE_Inp, mllib_ImageBlend_OMDA_SA_Inp, mllib_ImageBlend_OMDA_SAS_Inp, mllib_ImageBlend_OMDA_ZERO_Inp, mllib_ImageBlend_OMSA_DA_Inp, mllib_ImageBlend_OMSA_DC_Inp, mllib_ImageBlend_OMSA_OMDA_Inp, mllib_ImageBlend_OMSA_OMDC_Inp, mllib_ImageBlend_OMSA_OMSA_Inp, mllib_ImageBlend_OMSA_ONE_Inp, mllib_ImageBlend_OMSA_SA_Inp, mllib_ImageBlend_OMSA_SAS_Inp, mllib_ImageBlend_OMSA_ZERO_Inp, mllib_ImageBlend_OMSC_DA_Inp, mllib_ImageBlend_OMSC_DC_Inp, mllib_ImageBlend_OMSC_OMDA_Inp, mllib_ImageBlend_OMSC_OMDC_Inp, mllib_ImageBlend_OMSC_OMSA_Inp, mllib_ImageBlend_OMSC_ONE_Inp, mllib_ImageBlend_OMSC_SA_Inp, mllib_ImageBlend_OMSC_SAS_Inp, mllib_ImageBlend_OMSC_ZERO_Inp, mllib_ImageBlend_ONE_DA_Inp, mllib_ImageBlend_ONE_DC_Inp, mllib_ImageBlend_ONE_OMDA_Inp, mllib_ImageBlend_ONE_OMDC_Inp, mllib_ImageBlend_ONE_OMSA_Inp, mllib_ImageBlend_ONE_ONE_Inp, mllib_ImageBlend_ONE_SA_Inp, mllib_ImageBlend_ONE_SAS_Inp, mllib_ImageBlend_ONE_ZERO_Inp, mllib_ImageBlend_SA_DA_Inp, mllib_ImageBlend_SA_DC_Inp, mllib_ImageBlend_SA_OMDA_Inp, mllib_ImageBlend_SA_OMDC_Inp, mllib_ImageBlend_SA_OMSA_Inp, mllib_ImageBlend_SA_ONE_Inp, mllib_ImageBlend_SA_SA_Inp, mllib_ImageBlend_SA_SAS_Inp, mllib_ImageBlend_SA_ZERO_Inp, mllib_ImageBlend_SC_DA_Inp, mllib_ImageBlend_SC_DC_Inp, mllib_ImageBlend_SC_OMDA_Inp, mllib_ImageBlend_SC_OMDC_Inp, mllib_ImageBlend_SC_OMSA_Inp, mllib_ImageBlend_SC_ONE_Inp, mllib_ImageBlend_SC_SA_Inp, mllib_ImageBlend_SC_SAS_Inp, mllib_ImageBlend_SC_ZERO_Inp, mllib_ImageBlend_ZERO_DA_Inp, mllib_ImageBlend_ZERO_DC_Inp, mllib_ImageBlend_ZERO_OMDA_Inp, mllib_ImageBlend_ZERO_OMDC_Inp, mllib_ImageBlend_ZERO_OMSA_Inp, mllib_ImageBlend_ZERO_ONE_Inp, mllib_ImageBlend_ZERO_SA_Inp, mllib_ImageBlend_ZERO_SAS_Inp, mllib_ImageBlend_ZERO_ZERO_Inp – blending, in place
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageBlend_BSRC1_BSRC2_Inp(mllib_image *src1dst, const mllib_image *src2, mllib_s32 cmask);</pre>
DESCRIPTION	<p>This group of functions supports digital image composition. They are low-level, in-place, blending functions.</p> <p>The image type must be <code>MLIB_BYTE</code>. The input and output images must contain three or four channels. For three-channel images, the alpha value is as if the alpha value is 1.</p>

BSRC1 is one of the following: ZERO, ONE, SC, OMSC, DA, SA, OMDA, or OMSA.
 BSRC2 is one of the following: ZERO, ONE, DC, OMDC, DA, SA, OMDA, OMSA, or SAS.

The following are predefined blend factor types used in mediaLib image composition functions.

```
/* image blend factors */
typedef enum {
    MLIB_BLEND_ZERO,
    MLIB_BLEND_ONE,
    MLIB_BLEND_DST_COLOR,
    MLIB_BLEND_SRC_COLOR,
    MLIB_BLEND_ONE_MINUS_DST_COLOR,
    MLIB_BLEND_ONE_MINUS_SRC_COLOR,
    MLIB_BLEND_DST_ALPHA,
    MLIB_BLEND_SRC_ALPHA,
    MLIB_BLEND_ONE_MINUS_DST_ALPHA,
    MLIB_BLEND_ONE_MINUS_SRC_ALPHA,
    MLIB_BLEND_SRC_ALPHA_SATURATE
} mlib_blend;
```

See the following table for the definitions of the blend factors.

Type	Blend Factor [*]	Abbr.
MLIB_BLEND_ZERO	(0,0,0,0)	ZERO
MLIB_BLEND_ONE	(1,1,1,1)	ONE
MLIB_BLEND_DST_COLOR	(Rd,Gd,Bd,Ad)	DC
MLIB_BLEND_SRC_COLOR	(Rs,Gs,Bs,As)	SC
MLIB_BLEND_ONE_MINUS_DST_COLOR	(1,1,1,1)-(Rd,Gd,Bd,Ad)	OMDC
MLIB_BLEND_ONE_MINUS_SRC_COLOR	(1,1,1,1)-(Rs,Gs,Bs,As)	OMSC
MLIB_BLEND_DST_ALPHA	(Ad,Ad,Ad,Ad)	DA
MLIB_BLEND_SRC_ALPHA	(As,As,As,As)	SA
MLIB_BLEND_ONE_MINUS_DST_ALPHA	(1,1,1,1)-(Ad,Ad,Ad,Ad)	OMDA
MLIB_BLEND_ONE_MINUS_SRC_ALPHA	(1,1,1,1)-(As,As,As,As)	OMSA
MLIB_BLEND_SRC_ALPHA_SATURATE	(f,f,f,1)	SAS

[*]: The components of the first source image pixel are (Rd,Gd,Bd,Ad), and the components of the second source pixel are (Rs,Gs,Bs,As). Function $f = \min(As, 1-Ad)$. The first source image is also the destination image.

The blending formula for in-place processing is:

$$Cd = Cd * D + Cs * S$$

mllib_ImageBlend_BSRC1_BSRC2_Inp(3MLIB)

where C_d is the destination pixel (R_d, G_d, B_d, A_d), C_s is the source pixel (R_s, G_s, B_s, A_s), and D and S are the blend factors for the destination and source, respectively.

PARAMETERS Each of the functions takes the following arguments:

src1dst Pointer to the first source and the destination image.
src2 Pointer to the second source image.
cmask Channel mask to indicate the alpha channel. Each bit of the mask represents a channel in the image. The channel corresponding to the 1 bit is the alpha channel. *cmask* must be either 0x01 or 0x08.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageBlend_BSRC1_BSRC2(3MLIB)`, `mllib_ImageComposite(3MLIB)`, `mllib_ImageComposite_Inp(3MLIB)`, `attributes(5)`

NAME mllib_ImageBlendColor – blend an image and a color

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageBlendColor(mllib_image *dst, const
    mllib_image *src, const mllib_s32 *color, mllib_s32 cmask);
```

DESCRIPTION The mllib_ImageBlendColor() function blends an image and a color with the alpha channel.

It uses the following equation:

$$C_d = C_s * A_s + C_c * (1 - A_s)$$

$$A_d = 1.0$$

where, C_s and C_d are the RGB color components of the source and destination images, respectively. A_s and A_d are the alpha components of the source and destination images, respectively. C_c is the color component of the constant color.

For MLIB_BYTE images, the alpha coefficients are in Q8 format. For MLIB_SHORT images, the alpha coefficients are in Q15 format and must be positive. For MLIB_USHORT images, the alpha coefficients are in Q16 format. For MLIB_INT images, the alpha coefficients are in Q31 format and must be positive.

The images can have two to four channels. The length of color array must not be less than the number of channels in the images.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

color Array of constant color components.

cmask Channel mask to indicate the alpha channel. Each bit of cmask represents a channel in the image. The channel corresponding to the highest bit with value 1 is the alpha channel.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageBlendColor_Inp(3MLIB), mllib_ImageBlendColor_Fp(3MLIB), mllib_ImageBlendColor_Fp_Inp(3MLIB), attributes(5)

mllib_ImageBlendColor_Fp(3MLIB)

NAME	mllib_ImageBlendColor_Fp – blend an image and a color						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageBlendColor_Fp(mllib_image *dst, const mllib_image *src, const mllib_d64 *color, mllib_s32 cmask);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageBlendColor_Fp()</code> function blends an image and a color with the alpha channel.</p> <p>It uses the following equation:</p> $C_d = C_s * A_s + C_c * (1 - A_s)$ $A_d = 1.0$ <p>where, C_s and C_d are the RGB color components of the source and destination images, respectively. A_s and A_d are the alpha components of the source and destination images, respectively. C_c is the color component of the constant color.</p> <p>For <code>MLIB_FLOAT</code> and <code>MLIB_DOUBLE</code> images, the alpha coefficients are assumed to be in the range of [0.0, 1.0].</p> <p>The images can have two to four channels. The length of color array must not be less than the number of channels in the images.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>color</i> Array of constant color components.</p> <p><i>cmask</i> Channel mask to indicate the alpha channel. Each bit of <i>cmask</i> represents a channel in the image. The channel corresponding to the highest bit with value 1 is the alpha channel.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_ImageBlendColor_Fp_Inp(3MLIB) , mllib_ImageBlendColor(3MLIB) , mllib_ImageBlendColor_Inp(3MLIB) , attributes(5)						

mllib_ImageBlendColor_Fp_Inp(3MLIB)

NAME	mllib_ImageBlendColor_Fp_Inp – blend an image and a color, in place						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageBlendColor_Fp_Inp(mllib_image *<i>srcdst</i>, const mllib_d64 *<i>color</i>, mllib_s32 <i>cmask</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageBlendColor_Fp_Inp()</code> function blends an image and a color with the alpha channel.</p> <p>It uses the following equation:</p> $C_d = C_s * A_s + C_c * (1 - A_s)$ $A_d = 1.0$ <p>where, C_s and C_d are the RGB color components of the source and destination images, respectively. A_s and A_d are the alpha components of the source and destination images, respectively. C_c is the color component of the constant color.</p> <p>For <code>MLIB_FLOAT</code> and <code>MLIB_DOUBLE</code> images, the alpha coefficients are assumed to be in the range of [0.0, 1.0].</p> <p>The images can have two to four channels. The length of color array must not be less than the number of channels in the images.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>srcdst</i></td><td>Pointer to the source and destination image.</td></tr><tr><td><i>color</i></td><td>Array of constant color components.</td></tr><tr><td><i>cmask</i></td><td>Channel mask to indicate the alpha channel. Each bit of <i>cmask</i> represents a channel in the image. The channel corresponding to the highest bit with value 1 is the alpha channel.</td></tr></table>	<i>srcdst</i>	Pointer to the source and destination image.	<i>color</i>	Array of constant color components.	<i>cmask</i>	Channel mask to indicate the alpha channel. Each bit of <i>cmask</i> represents a channel in the image. The channel corresponding to the highest bit with value 1 is the alpha channel.
<i>srcdst</i>	Pointer to the source and destination image.						
<i>color</i>	Array of constant color components.						
<i>cmask</i>	Channel mask to indicate the alpha channel. Each bit of <i>cmask</i> represents a channel in the image. The channel corresponding to the highest bit with value 1 is the alpha channel.						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageBlendColor_Fp(3MLIB)</code> , <code>mllib_ImageBlendColor(3MLIB)</code> , <code>mllib_ImageBlendColor_Inp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageBlendColor_Inp(3MLIB)

NAME	mllib_ImageBlendColor_Inp – blend an image and a color, in place						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageBlendColor_Inp(mllib_image *srcdst, const mllib_s32 *color, mllib_s32 cmask);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageBlendColor_Inp()</code> function blends an image and a color with the alpha channel.</p> <p>It uses the following equation:</p> $C_d = C_s * A_s + C_c * (1 - A_s)$ $A_d = 1.0$ <p>where, C_s and C_d are the RGB color components of the source and destination images, respectively. A_s and A_d are the alpha components of the source and destination images, respectively. C_c is the color component of the constant color.</p> <p>For <code>MLIB_BYTE</code> images, the alpha coefficients are in Q8 format. For <code>MLIB_SHORT</code> images, the alpha coefficients are in Q15 format and must be positive. For <code>MLIB_USHORT</code> images, the alpha coefficients are in Q16 format. For <code>MLIB_INT</code> images, the alpha coefficients are in Q31 format and must be positive.</p> <p>The images can have two to four channels. The length of color array must not be less than the number of channels in the images.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>srcdst</i></td><td>Pointer to the source and destination image.</td></tr><tr><td><i>color</i></td><td>Array of constant color components.</td></tr><tr><td><i>cmask</i></td><td>Channel mask to indicate the alpha channel. Each bit of <i>cmask</i> represents a channel in the image. The channel corresponding to the highest bit with value 1 is the alpha channel.</td></tr></table>	<i>srcdst</i>	Pointer to the source and destination image.	<i>color</i>	Array of constant color components.	<i>cmask</i>	Channel mask to indicate the alpha channel. Each bit of <i>cmask</i> represents a channel in the image. The channel corresponding to the highest bit with value 1 is the alpha channel.
<i>srcdst</i>	Pointer to the source and destination image.						
<i>color</i>	Array of constant color components.						
<i>cmask</i>	Channel mask to indicate the alpha channel. Each bit of <i>cmask</i> represents a channel in the image. The channel corresponding to the highest bit with value 1 is the alpha channel.						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageBlendColor(3MLIB)</code> , <code>mllib_ImageBlendColor_Fp(3MLIB)</code> , <code>mllib_ImageBlendColor_Fp_Inp(3MLIB)</code> , <code>attributes(5)</code>						

NAME	mlib_ImageBlend_Fp – blend with an alpha image						
SYNOPSIS	<pre>cc [flag...] file... -lmlib [library...] #include <mlib.h> mlib_status mlib_ImageBlend_Fp(mlib_image *dst, const mlib_image *src1, const mlib_image *src2, const mlib_image *alpha);</pre>						
DESCRIPTION	<p>The <code>mlib_ImageBlend_Fp()</code> function blends two images together on a pixel-by-pixel basis using an alpha image, when alpha is also on a pixel basis. The <i>alpha</i> image can be a single-channel image or have the same number of channels as the source and destination images.</p> <p>It uses the following equation when the <i>alpha</i> image is a single-channel image:</p> $dst[x][y][i] = alpha[x][y][0]*src1[x][y][i] + (1 - alpha[x][y][0])*src2[x][y][i]$ <p>It uses the following equation when the <i>alpha</i> image has the same number of channels as the source and destination images:</p> $dst[x][y][i] = alpha[x][y][i]*src1[x][y][i] + (1 - alpha[x][y][i])*src2[x][y][i]$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src1</i> Pointer to first source image.</p> <p><i>src2</i> Pointer to second source image.</p> <p><i>alpha</i> Alpha image used to control blending. The pixels in this image should have values in the range of [0.0, 1.0].</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mlib_ImageBlend(3MLIB)</code> , <code>mlib_ImageBlend1_Fp_Inp(3MLIB)</code> , <code>mlib_ImageBlend1_Inp(3MLIB)</code> , <code>mlib_ImageBlend2_Fp_Inp(3MLIB)</code> , <code>mlib_ImageBlend2_Inp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageBlendMulti(3MLIB)

NAME	mllib_ImageBlendMulti – blend multiple images
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageBlendMulti(mllib_image *dst, const mllib_image **srcs, const mllib_image **alphas, const mllib_s32 *c, mllib_s32 n);</pre>
DESCRIPTION	<p>The <code>mllib_ImageBlendMulti()</code> function blends multiple source images, using multiple alpha images, into a single destination image.</p> <p>All images involved should have the same data type and same size and the source and destination images should have the same number of channels. The alpha images should have either 1 channel or the same number of channels as the sources and destination. A single-channel alpha image would be applied to all channels of the corresponding source image. Single and multi-channel alpha images can be mixed in the same invocation.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \frac{\sum_{k=0}^{n-1} \{alphas[k][x][y][j] * srcs[k][x][y][i]\}}{\sum_{k=0}^{n-1} \{alphas[k][x][y][j]\}}$ <p>or</p> $dst[x][y][i] = c[i] \quad \text{if } \sum_{k=0}^{n-1} \{alphas[k][x][y][j]\} = 0$ <p>where $j = i$ for multi-channel alpha images; $j = 0$ for single-channel alpha images.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>srcs</i> Pointer to an array of source images.</p> <p><i>alphas</i> Pointer to an array of alpha images.</p> <p><i>c</i> Background color.</p> <p><i>n</i> Number of source images to be blended.</p>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .

mllib_ImageBlendMulti(3MLIB)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageBlendMulti_Fp(3MLIB)`, `mllib_ImageBlend(3MLIB)`,
`mllib_ImageBlend_Fp(3MLIB)`, `attributes(5)`

mllib_ImageBlendMulti_Fp(3MLIB)

NAME	mllib_ImageBlendMulti_Fp – blend multiple images										
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageBlendMulti_Fp(mllib_image *dst, const mllib_image **srcs, const mllib_image **alphas, const mllib_d64 *c, mllib_s32 n);</pre>										
DESCRIPTION	<p>The <code>mllib_ImageBlendMulti_Fp()</code> function blends multiple source images, using multiple alpha images, into a single destination image.</p> <p>All images involved should have the same data type and same size and the source and destination images should have the same number of channels. The alpha images should have either 1 channel or the same number of channels as the sources and destination. A single-channel alpha image would be applied to all channels of the corresponding source image. Single and multi-channel alpha images can be mixed in the same invocation.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \frac{\sum_{k=0}^{n-1} \{alphas[k][x][y][j] * srcs[k][x][y][i]\}}{\sum_{k=0}^{n-1} \{alphas[k][x][y][j]\}}$ <p>or</p> $dst[x][y][i] = c[i] \quad \text{if } \sum_{k=0}^{n-1} \{alphas[k][x][y][j]\} = 0$ <p>where $j = i$ for multi-channel alpha images; $j = 0$ for single-channel alpha images.</p>										
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>dst</i></td><td>Pointer to destination image.</td></tr><tr><td><i>srcs</i></td><td>Pointer to an array of source images.</td></tr><tr><td><i>alphas</i></td><td>Pointer to an array of alpha images.</td></tr><tr><td><i>c</i></td><td>Background color.</td></tr><tr><td><i>n</i></td><td>Number of source images to be blended.</td></tr></table>	<i>dst</i>	Pointer to destination image.	<i>srcs</i>	Pointer to an array of source images.	<i>alphas</i>	Pointer to an array of alpha images.	<i>c</i>	Background color.	<i>n</i>	Number of source images to be blended.
<i>dst</i>	Pointer to destination image.										
<i>srcs</i>	Pointer to an array of source images.										
<i>alphas</i>	Pointer to an array of alpha images.										
<i>c</i>	Background color.										
<i>n</i>	Number of source images to be blended.										
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .										

`mlib_ImageBlendMulti_Fp(3MLIB)`

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mlib_ImageBlendMulti(3MLIB)`, `mlib_ImageBlend(3MLIB)`,
`mlib_ImageBlend_Fp(3MLIB)`, `attributes(5)`

mllib_ImageBlendRGBA2ARGB(3MLIB)

NAME	mllib_ImageBlendRGBA2ARGB – image blending and channel reordering						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageBlendRGBA2ARGB(mllib_image *dst, const mllib_image *src);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageBlendRGBA2ARGB()</code> function blends the source image of the RGBA format into the destination image of the ARGB format.</p> <p>The image type must be <code>MLIB_BYTE</code>. The source and destination images must contain four channels.</p> <p>It uses the following equation:</p> $C_d = C_s * A_s + C_d * (1 - A_s)$ $A_d = A_d$ <p>where, C_s and C_d are the RGB color components of the source and destination images, respectively. A_s and A_d are the alpha components of the source and destination images, respectively.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageBlendRGBA2BGRA(3MLIB)</code> , <code>mllib_ImageBlend_OMSA_SA_Inp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageBlendRGBA2BGRA(3MLIB)

NAME	mllib_ImageBlendRGBA2BGRA – image blending and channel reordering						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageBlendRGBA2BGRA(mllib_image *dst, const mllib_image *src);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageBlendRGBA2BGRA()</code> function blends the source image of the RGBA format into the destination image of the BGRA format.</p> <p>The image type must be <code>MLIB_BYTE</code>. The source and destination images must contain four channels.</p> <p>It uses the following equation:</p> $C_d = C_s * A_s + C_d * (1 - A_s)$ $A_d = A_d$ <p>where, C_s and C_d are the RGB color components of the source and destination images, respectively. A_s and A_d are the alpha components of the source and destination images, respectively.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageBlendRGBA2ARGB(3MLIB)</code> , <code>mllib_ImageBlend_OMSA_SA_Inp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageChannelCopy(3MLIB)

NAME | mllib_ImageChannelCopy – channel copy

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>
```



```
mllib_status mllib_ImageChannelCopy(mllib_image *dst, const  
mllib_image *src, mllib_s32 cmask);
```

DESCRIPTION | The `mllib_ImageChannelCopy()` function copies the selected channels of the source image into the corresponding channels of the destination image. The data type of the image can be `MLIB_BYTE`, `MLIB_SHORT`, `MLIB_USHORT`, `MLIB_INT`, `MLIB_FLOAT`, or `MLIB_DOUBLE`.

PARAMETERS | The function takes the following arguments:

<i>dst</i>	Pointer to a destination image.
<i>src</i>	Pointer to a source image.
<i>cmask</i>	Source or destination channel selection mask. Each bit of the mask represents a channel in the image data. The least significant bit (LSB) of the mask corresponds to the last channel in the image data. A bit with a value of 1 indicates that the channel is selected.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageChannelExtract(3MLIB)`, `mllib_ImageChannelInsert(3MLIB)`, `mllib_ImageChannelMerge(3MLIB)`, `mllib_ImageChannelSplit(3MLIB)`, `attributes(5)`

NAME mllib_ImageChannelExtract – channel extract

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageChannelExtract(mllib_image *dst, const
mllib_image *src, mllib_s32 cmask);
```

DESCRIPTION In the `mllib_ImageChannelExtract()` function, the selected N channels in the source image are copied into the destination image, where N is the number of channels in the destination image. If more than N channels are selected, then the leftmost N channels are extracted. If less than N channels are selected, then the function returns failure status. The channel mask is defined with respect to the source image. The data type of the image can be `MLIB_BYTE`, `MLIB_SHORT`, `MLIB_USHORT`, `MLIB_INT`, `MLIB_FLOAT`, or `MLIB_DOUBLE`.

PARAMETERS The function takes the following arguments:

dst Pointer to a destination image.

src Pointer to a source image.

cmask Source or destination channel selection mask. Each bit of the mask represents a channel in the image data. The least significant bit (LSB) of the mask corresponds to the last channel in the image data. A bit with a value of 1 indicates that the channel is selected.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageChannelCopy(3MLIB)`, `mllib_ImageChannelInsert(3MLIB)`, `mllib_ImageChannelMerge(3MLIB)`, `mllib_ImageChannelSplit(3MLIB)`, `attributes(5)`

mllib_ImageChannelInsert(3MLIB)

NAME	mllib_ImageChannelInsert – channel insert						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageChannelInsert(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, mllib_s32 <i>cmask</i>);</pre>						
DESCRIPTION	In the <code>mllib_ImageChannelInsert()</code> function, all N channels in the source image are copied into the selected channels in the destination image, where N is the number of channels in the source image. If more than N channels are selected, then the leftmost N channels are inserted. If less than N channels are selected, then the function returns failure status. The channel mask is defined with respect to the destination image. The data type of the image can be <code>MLIB_BYTE</code> , <code>MLIB_SHORT</code> , <code>MLIB_USHORT</code> , <code>MLIB_INT</code> , <code>MLIB_FLOAT</code> , or <code>MLIB_DOUBLE</code> .						
PARAMETERS	The function takes the following arguments: <table><tr><td><i>dst</i></td><td>Pointer to a destination image.</td></tr><tr><td><i>src</i></td><td>Pointer to a source image.</td></tr><tr><td><i>cmask</i></td><td>Source or destination channel selection mask. Each bit of the mask represents a channel in the image data. The least significant bit (LSB) of the mask corresponds to the last channel in the image data. A bit with a value of 1 indicates that the channel is selected.</td></tr></table>	<i>dst</i>	Pointer to a destination image.	<i>src</i>	Pointer to a source image.	<i>cmask</i>	Source or destination channel selection mask. Each bit of the mask represents a channel in the image data. The least significant bit (LSB) of the mask corresponds to the last channel in the image data. A bit with a value of 1 indicates that the channel is selected.
<i>dst</i>	Pointer to a destination image.						
<i>src</i>	Pointer to a source image.						
<i>cmask</i>	Source or destination channel selection mask. Each bit of the mask represents a channel in the image data. The least significant bit (LSB) of the mask corresponds to the last channel in the image data. A bit with a value of 1 indicates that the channel is selected.						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageChannelCopy(3MLIB)</code> , <code>mllib_ImageChannelExtract(3MLIB)</code> , <code>mllib_ImageChannelMerge(3MLIB)</code> , <code>mllib_ImageChannelSplit(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageChannelMerge – channel merge

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageChannelMerge(mllib_image *dst, const
    mllib_image **srcs);
```

DESCRIPTION The mllib_ImageChannelMerge() function converts an array of single-channel images into a multi-channel image.

A0 A1 A2 ...
 B0 B1 B2 ... ==> A0 B0 C0 A1 B1 C1 A2 B2 C2 ...
 C0 C1 C2 ...

All images must have the same type, same width, and same height. The data type of the images can be MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, MLIB_INT, MLIB_FLOAT, or MLIB_DOUBLE. The destination image must have the number of channels equal to the number of images in the *srcs* array. The source images must be single-channel images.

PARAMETERS The function takes the following arguments:

dst Pointer to a multi-channel destination image.

srcs Pointer to an array of single-channel source images.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageChannelCopy(3MLIB), mllib_ImageChannelExtract(3MLIB), mllib_ImageChannelInsert(3MLIB), mllib_ImageChannelSplit(3MLIB), attributes(5)

mllib_ImageChannelSplit(3MLIB)

NAME	mllib_ImageChannelSplit – channel split						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageChannelSplit(mllib_image **<i>dsts</i>, const mllib_image *<i>src</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageChannelSplit()</code> function converts a multi-channel image into an array of single-channel images.</p> <pre> A0 A1 A2 ... A0 B0 C0 A1 B1 C1 A2 B2 C2 ... ==> B0 B1 B2 ... C0 C1 C2 ...</pre> <p>All images must have the same type, same width, and same height. The data type of the images can be <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, <code>MLIB_INT</code>, <code>MLIB_FLOAT</code>, or <code>MLIB_DOUBLE</code>. The source image must have the number of channels equal to the number of images in the <code>dsts</code> array. The destination images must be single-channel images.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dsts</i> Pointer to an array of single-channel destination images.</p> <p><i>src</i> Pointer to a multi-channel source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageChannelCopy(3MLIB)</code> , <code>mllib_ImageChannelExtract(3MLIB)</code> , <code>mllib_ImageChannelInsert(3MLIB)</code> , <code>mllib_ImageChannelMerge(3MLIB)</code> , <code>attributes(5)</code>						

NAME	mllib_ImageClear – clear						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageClear(mllib_image *<i>img</i>, const mllib_s32 *<i>color</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageClear()</code> function sets an image to a specific color. The data type of the image can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p> <p>It uses the following equation:</p> $\text{img}[x][y][i] = \text{color}[i]$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>img</i> Pointer to an image.</p> <p><i>color</i> Array of color values by channel. <code>color[i]</code> contains the value for channel <i>i</i>.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageClear_Fp(3MLIB)</code> , <code>mllib_ImageClearEdge(3MLIB)</code> , <code>mllib_ImageClearEdge_Fp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageClearEdge(3MLIB)

NAME mllib_ImageClearEdge – sets edges of an image to a specific color

SYNOPSIS
cc [*flag...*] *file...* -lmllib [*library...*]
#include <mllib.h>

```
mllib_status mllib_ImageClearEdge(mllib_image *img, mllib_s32 dx,  
    mllib_s32 dy, const mllib_s32 *color);
```

DESCRIPTION The mllib_ImageClearEdge() function sets edges of an image to a specific color. This function can be used in conjunction with the convolve and other spatial functions to fill in the pixel values along the edges. The data type of the image can be MLIB_BIT, MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, or MLIB_INT.

PARAMETERS The function takes the following arguments:

<i>img</i>	Pointer to an image.
<i>dx</i>	Number of columns on the left and right edges of the image to be cleared.
<i>dy</i>	Number of rows at the top and bottom edges of the image to be cleared.
<i>color</i>	Array of color values by channel. <i>color</i> [<i>i</i>] contains the value for channel <i>i</i> .

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageClear(3MLIB), mllib_ImageClear_Fp(3MLIB), mllib_ImageClearEdge_Fp(3MLIB), attributes(5)

NAME mllib_ImageClearEdge_Fp – sets edges of an image to a specific color

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageClearEdge_Fp(mllib_image *img, mllib_s32 dx,
    mllib_s32 dy, const mllib_d64 *color);
```

DESCRIPTION The mllib_ImageClearEdge_Fp() function sets edges of an image to a specific color. This function can be used in conjunction with the convolve and other spatial functions to fill in the pixel values along the edges. The data type of the image can be MLIB_FLOAT or MLIB_DOUBLE.

PARAMETERS The function takes the following arguments:

img Pointer to an image.

dx Number of columns on the left and right edges of the image to be cleared.

dy Number of rows at the top and bottom edges of the image to be cleared.

color Array of color values by channel. color[i] contains the value for channel i.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageClear(3MLIB), mllib_ImageClearEdge(3MLIB), mllib_ImageClearEdge_Fp(3MLIB), attributes(5)

mllib_ImageClear_Fp(3MLIB)

NAME mllib_ImageClear_Fp – clear

SYNOPSIS
cc [*flag...*] *file...* -lmllib [*library...*]
#include <mllib.h>

```
mllib_status mllib_ImageClear_Fp(mllib_image *img, const mllib_d64  
    *color);
```

DESCRIPTION The `mllib_ImageClear_Fp()` function sets an image to a specific color. The data type of the image can be `MLIB_FLOAT` or `MLIB_DOUBLE`.

It uses the following equation:

$$\text{img}[x][y][i] = \text{color}[i]$$

PARAMETERS The function takes the following arguments:

img Pointer to an image.

color Array of color values by channel. `color[i]` contains the value for channel *i*.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageClear(3MLIB)`, `mllib_ImageClearEdge(3MLIB)`, `mllib_ImageClearEdge_Fp(3MLIB)`, `attributes(5)`

mllib_ImageColorConvert1(3MLIB)

NAME | mllib_ImageColorConvert1 – color conversion using a 3x3 floating-point matrix

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageColorConvert1(mllib_image *dst, const
    mllib_image *src, const mllib_d64 *cmat);
```

DESCRIPTION | The mllib_ImageColorConvert1() function takes a 3x3 floating-point conversion matrix and converts the source color image to the destination color image.

The source and destination images must be three-channel images.

It uses the following equation:

$$\begin{array}{|c|c|c|c|} \hline |dst[x][y][0]| & |cmat[0] cmat[1] cmat[2]| & |src[x][y][0]| \\ |dst[x][y][1]| & = |cmat[3] cmat[4] cmat[5]| * & |src[x][y][1]| \\ |dst[x][y][2]| & |cmat[6] cmat[7] cmat[8]| & |src[x][y][2]| \\ \hline \end{array}$$

PARAMETERS | The function takes the following arguments:

dst | Pointer to destination image.

src | Pointer to source image.

cmat | Conversion matrix in row major order.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_ImageColorConvert1_Fp(3MLIB), mllib_ImageColorConvert2(3MLIB), mllib_ImageColorConvert2_Fp(3MLIB), mllib_ImageColorRGB2XYZ(3MLIB), mllib_ImageColorRGB2XYZ_Fp(3MLIB), mllib_ImageColorXYZ2RGB(3MLIB), mllib_ImageColorXYZ2RGB_Fp(3MLIB), attributes(5)

mllib_ImageColorConvert1_Fp(3MLIB)

NAME	mllib_ImageColorConvert1_Fp – color conversion using a 3x3 floating-point matrix						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageColorConvert1_Fp(mllib_image *dst, const mllib_image *src, const mllib_d64 *cmat);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageColorConvert1_Fp()</code> function takes a 3x3 floating point conversion matrix and converts the floating-point source color image to the destination color image.</p> <p>The source and destination images must be three-channel images.</p> <p>It uses the following equation:</p> $\begin{array}{ c c c } \hline dst[x][y][0] & cmat[0] cmat[1] cmat[2] & src[x][y][0] \\ \hline dst[x][y][1] & = cmat[3] cmat[4] cmat[5] * & src[x][y][1] \\ \hline dst[x][y][2] & cmat[6] cmat[7] cmat[8] & src[x][y][2] \\ \hline \end{array}$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>cmat</i> Conversion matrix in row major order.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageColorConvert1(3MLIB)</code> , <code>mllib_ImageColorConvert2(3MLIB)</code> , <code>mllib_ImageColorConvert2_Fp(3MLIB)</code> , <code>mllib_ImageColorRGB2XYZ(3MLIB)</code> , <code>mllib_ImageColorRGB2XYZ_Fp(3MLIB)</code> , <code>mllib_ImageColorXYZ2RGB(3MLIB)</code> , <code>mllib_ImageColorXYZ2RGB_Fp(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageColorConvert2 – color conversion using a 3x3 floating-point matrix and a three-element offset

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageColorConvert2(mllib_image *dst, const
    mllib_image *src, const mllib_d64 *cmat, const mllib_d64 *offset);
```

DESCRIPTION

The `mllib_ImageColorConvert2()` function takes a 3x3 floating-point conversion matrix and a three-element offset and converts the source color image to the destination color image.

The source and destination images must be three-channel images.

It uses the following equation:

$$\begin{array}{|c|} \hline |dst[x][y][0]| \\ \hline |dst[x][y][1]| \\ \hline |dst[x][y][2]| \\ \hline \end{array} = \begin{array}{|c|} \hline |cmat[0] cmat[1] cmat[2]| \\ \hline |cmat[3] cmat[4] cmat[5]| \\ \hline |cmat[6] cmat[7] cmat[8]| \\ \hline \end{array} * \begin{array}{|c|} \hline |src[x][y][0]| \\ \hline |src[x][y][1]| \\ \hline |src[x][y][2]| \\ \hline \end{array} + \begin{array}{|c|} \hline |offset[0]| \\ \hline |offset[1]| \\ \hline |offset[2]| \\ \hline \end{array}$$

PARAMETERS

The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

cmat Conversion matrix in row major order.

offset Offset array.

RETURN VALUES

The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageColorConvert1(3MLIB)`, `mllib_ImageColorConvert1_Fp(3MLIB)`, `mllib_ImageColorConvert2_Fp(3MLIB)`, `mllib_ImageColorRGB2YCC(3MLIB)`, `mllib_ImageColorRGB2YCC_Fp(3MLIB)`, `mllib_ImageColorYCC2RGB(3MLIB)`, `mllib_ImageColorYCC2RGB_Fp(3MLIB)`, `attributes(5)`

mllib_ImageColorConvert2_Fp(3MLIB)

NAME	mllib_ImageColorConvert2_Fp – color conversion using a 3x3 floating-point matrix and a three-element offset						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageColorConvert2_Fp(mllib_image *dst, const mllib_image *src, const mllib_d64 *cmat, const mllib_d64 *offset);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageColorConvert2_Fp()</code> function takes a 3x3 floating-point conversion matrix and a three-element offset and converts the floating-point source color image to the destination color image.</p> <p>The source and destination images must be three-channel images.</p> <p>It uses the following equation:</p> $\begin{array}{ c c c } \hline dst[x][y][0] & cmat[0] cmat[1] cmat[2] & src[x][y][0] & offset[0] \\ \hline dst[x][y][1] & = cmat[3] cmat[4] cmat[5] * & src[x][y][1] & + offset[1] \\ \hline dst[x][y][2] & cmat[6] cmat[7] cmat[8] & src[x][y][2] & offset[2] \\ \hline \end{array}$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>cmat</i> Conversion matrix in row major order.</p> <p><i>offset</i> Offset array.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageColorConvert1(3MLIB)</code> , <code>mllib_ImageColorConvert1_Fp(3MLIB)</code> , <code>mllib_ImageColorConvert2(3MLIB)</code> , <code>mllib_ImageColorRGB2YCC(3MLIB)</code> , <code>mllib_ImageColorRGB2YCC_Fp(3MLIB)</code> , <code>mllib_ImageColorYCC2RGB(3MLIB)</code> , <code>mllib_ImageColorYCC2RGB_Fp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageColorDitherFree(3MLIB)

NAME mllib_ImageColorDitherFree – release the internal data structure for image dithering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_ImageColorDitherFree(void *colormap);
```

DESCRIPTION The mllib_ImageColorDitherFree() function releases an internal data structure, *colormap*, which was created by mllib_ImageColorDitherInit() and was used by one of the following functions for image dithering:

mllib_ImageColorErrorDiffusion3x3
mllib_ImageColorErrorDiffusionMxN
mllib_ImageColorOrderedDither8x8
mllib_ImageColorOrderedDitherMxN

PARAMETERS The function takes the following arguments:

colormap Internal data structure for image dithering.

RETURN VALUES None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageColorDitherInit(3MLIB),
mllib_ImageColorErrorDiffusion3x3(3MLIB),
mllib_ImageColorErrorDiffusionMxN(3MLIB),
mllib_ImageColorOrderedDither8x8(3MLIB),
mllib_ImageColorOrderedDitherMxN(3MLIB), attributes(5)

mllib_ImageColorDitherInit(3MLIB)

NAME	mllib_ImageColorDitherInit – initialization for image dithering																				
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageColorDitherInit(void **colormap, const mllib_s32 *dimensions, mllib_type intype, mllib_type outtype, mllib_s32 channels, mllib_s32 entries, mllib_s32 offset, void **lut);</pre>																				
DESCRIPTION	<p>The <code>mllib_ImageColorDitherInit()</code> function creates an internal data structure, <code>colormap</code>, which can be used by one of the following functions for image dithering:</p> <pre>mllib_ImageColorErrorDiffusion3x3 mllib_ImageColorErrorDiffusionMxN mllib_ImageColorOrderedDither8x8 mllib_ImageColorOrderedDitherMxN</pre> <p>The <code>lut</code> might have either 1 or 3 channels. The type of the <code>lut</code> can be one of the following:</p> <pre>MLIB_BYTE in, MLIB_BYTE out (i.e., BYTE-to-BYTE) MLIB_BIT in, MLIB_BYTE out (i.e., BIT-to-BYTE)</pre> <p>If <code>dimensions == NULL</code>, then no colorcube will be created. In this case, the user-provided lookup table, <code>lut</code>, will be used for dithering.</p> <p>If <code>dimensions != NULL</code>, then a colorcube is created from scratch in a way shown in the following example.</p> <p>To dither an RGB image of type <code>MLIB_BYTE</code> to a color-indexed image of type <code>MLIB_BYTE</code>, we can use the following parameters:</p> <pre>mllib_s32 dimensions[] = {2, 3, 4}; mllib_type intype = MLIB_BYTE; mllib_type outtype = MLIB_BYTE; mllib_s32 channels = 3; mllib_s32 offset = 6;</pre> <p>These values would lead to the creation of a colorcube that would dither red values in the source image to one of 2 red levels, green values to one of 3 green levels, and blue values to one of 4 blue levels. You could picture this colorcube as a cube with dimensions of 2, 3, and 4. The index values assigned to the elements in that cube can be described by the following lookup table:</p> <table><thead><tr><th>Indexes</th><th>Red Values</th><th>Green Values</th><th>Blue Values</th></tr></thead><tbody><tr><td>0</td><td></td><td></td><td></td></tr><tr><td>...</td><td></td><td></td><td></td></tr><tr><td>5</td><td></td><td></td><td></td></tr><tr><td>6</td><td>0</td><td>0</td><td>0</td></tr></tbody></table>	Indexes	Red Values	Green Values	Blue Values	0				...				5				6	0	0	0
Indexes	Red Values	Green Values	Blue Values																		
0																					
...																					
5																					
6	0	0	0																		

mllib_ImageColorDitherInit(3MLIB)

Indexes	Red Values	Green Values	Blue Values
7	255	0	0
8	0	128	0
9	255	128	0
10	0	255	0
11	255	255	0
12	0	0	85
13	255	0	85
14	0	128	85
15	255	128	85
16	0	255	85
17	255	255	85
18	0	0	170
19	255	0	170
20	0	128	170
21	255	128	170
22	0	255	170
23	255	255	170
24	0	0	255
25	255	0	255
26	0	128	255
27	255	128	255
28	0	255	255
29	255	255	255
...			

The distance between level changes in each channel of the lookup table is determined by the following formulas:

```

multipliers[0] = sizeof(dimensions[0])*1;
multipliers[i] = sizeof(dimensions[i])*
                abs(multipliers[i-1]*dimension[i-1]);

```

mllib_ImageColorDitherInit(3MLIB)

A negative `dimensions[i]`, so as to a negative `multipliers[i]`, indicates that the values in a color ramp for channel `i` should appear in decreasing as opposed to increasing order.

For each channel `i`, the values of the levels are determined by the following formulas:

```
double delta = (dataMax - dataMin) / (abs(dimensions[i]) - 1);  
int levels[j] = (int)(j*delta + 0.5);
```

where `dataMax` and `dataMin` are the maximum and minimum values, respectively, for data type *intype*.

Whenever a colorcube is created, if `lut != NULL`, the lookup table will be filled according to the colorcube and supplied parameters like *offset*. For the example shown above, the lookup table will start from line 6. In this case, it is the user's responsibility to allocate memory for the lookup table.

PARAMETERS

The function takes the following arguments:

<i>colormap</i>	Internal data structure for image dithering.
<i>dimensions</i>	Dimensions of the colorcube in the colormap structure.
<i>intype</i>	Data type of the source image and the lookup table.
<i>outtype</i>	Data type of the destination indexed image.
<i>channels</i>	Number of channels of the lookup table and source image.
<i>entries</i>	Number of entries of the lookup table.
<i>offset</i>	Index offset of the lookup table.
<i>lut</i>	Lookup table.

RETURN VALUES

The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

`mllib_ImageColorDitherFree(3MLIB)`,
`mllib_ImageColorErrorDiffusion3x3(3MLIB)`,
`mllib_ImageColorErrorDiffusionMxN(3MLIB)`,
`mllib_ImageColorOrderedDither8x8(3MLIB)`,
`mllib_ImageColorOrderedDitherMxN(3MLIB)`, `attributes(5)`

mllib_ImageColorErrorDiffusion3x3(3MLIB)

NAME mllib_ImageColorErrorDiffusion3x3 – true color to indexed color conversion using error diffusion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageColorErrorDiffusion3x3(mllib_image *dst,
    const mllib_image *src, const mllib_s32 *kernel, mllib_s32 scale,
    const void *colormap);
```

DESCRIPTION The `mllib_ImageColorErrorDiffusion3x3()` function converts a true color image to a pseudo color image with the method of error diffusion dithering. The source image can be an `MLIB_BYTE` or `MLIB_SHORT` image with three or four channels. The destination must be a single-channel `MLIB_BYTE` or `MLIB_SHORT` image.

The last parameter, `colormap`, is an internal data structure for inverse color mapping. Create it by calling the `mllib_ImageColorTrue2IndexInit()` function.

PARAMETERS The function takes the following arguments:

- dst* Pointer to destination or destination image.
- src* Pointer to source or source image.
- kernel* Pointer to the 3x3 error-distribution kernel, in row major order.
- scale* The scaling factor for kernel to convert the input integer coefficients into floating-point coefficients:

floating-point coefficient = integer coefficient * 2**(-scale)
- colormap* Internal data structure for inverse color mapping.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageColorOrderedDither8x8(3MLIB)`,
`mllib_ImageColorTrue2Index(3MLIB)`,
`mllib_ImageColorTrue2IndexFree(3MLIB)`,
`mllib_ImageColorTrue2IndexInit(3MLIB)`, `attributes(5)`

mllib_ImageColorErrorDiffusionMxN(3MLIB)

NAME	mllib_ImageColorErrorDiffusionMxN – true-color to indexed-color or grayscale to black-white conversion, using error diffusion																		
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageColorErrorDiffusionMxN(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, const mllib_s32 *<i>kernel</i>, mllib_s32 <i>m</i>, mllib_s32 <i>n</i>, mllib_s32 <i>dm</i>, mllib_s32 <i>dn</i>, mllib_s32 <i>scale</i>, const void *<i>colormap</i>);</pre>																		
DESCRIPTION	<p>The <code>mllib_ImageColorErrorDiffusionMxN()</code> function converts a 3-channel image to a 1-channel indexed image, or converts a 1-channel grayscale image to a 1-channel MLIB_BIT image, with the method of error diffusion.</p> <p>The <i>src</i> can be an MLIB_BYTE image with 1 or 3 channels. The <i>dst</i> must be a 1-channel MLIB_BIT or MLIB_BYTE image.</p> <p>The <i>colormap</i> must be created by <code>mllib_ImageColorDitherInit()</code>. It may or may not have a colorcube included. If it does, the colorcube is used. Otherwise, the general lookup table included in <i>colormap</i> is used.</p> <p>The kernel is required to have the following property:</p> <pre>kernel[0] = kernel[1] = ... = kernel[m*dn+dm] = 0; kernel[m*dn+dm+1] + ... + kernel[m*n-1] = 2**scale; scale ≥ 0</pre>																		
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>dst</i></td><td>Pointer to destination image.</td></tr><tr><td><i>src</i></td><td>Pointer to source image.</td></tr><tr><td><i>kernel</i></td><td>Pointer to the error-distribution kernel, in row major order.</td></tr><tr><td><i>m</i></td><td>Kernel width. $m > 1$.</td></tr><tr><td><i>n</i></td><td>Kernel height. $n > 1$.</td></tr><tr><td><i>dm</i></td><td>X coordinate of the key element in the kernel. $0 \leq dm < m$.</td></tr><tr><td><i>dn</i></td><td>Y coordinate of the key element in the kernel. $0 \leq dn < n$.</td></tr><tr><td><i>scale</i></td><td>The scaling factor for kernel to convert the input integer coefficients into floating-point coefficients: floating-point coefficient = integer coefficient * 2**(-scale)</td></tr><tr><td><i>colormap</i></td><td>Internal data structure for image dithering.</td></tr></table>	<i>dst</i>	Pointer to destination image.	<i>src</i>	Pointer to source image.	<i>kernel</i>	Pointer to the error-distribution kernel, in row major order.	<i>m</i>	Kernel width. $m > 1$.	<i>n</i>	Kernel height. $n > 1$.	<i>dm</i>	X coordinate of the key element in the kernel. $0 \leq dm < m$.	<i>dn</i>	Y coordinate of the key element in the kernel. $0 \leq dn < n$.	<i>scale</i>	The scaling factor for kernel to convert the input integer coefficients into floating-point coefficients: floating-point coefficient = integer coefficient * 2**(-scale)	<i>colormap</i>	Internal data structure for image dithering.
<i>dst</i>	Pointer to destination image.																		
<i>src</i>	Pointer to source image.																		
<i>kernel</i>	Pointer to the error-distribution kernel, in row major order.																		
<i>m</i>	Kernel width. $m > 1$.																		
<i>n</i>	Kernel height. $n > 1$.																		
<i>dm</i>	X coordinate of the key element in the kernel. $0 \leq dm < m$.																		
<i>dn</i>	Y coordinate of the key element in the kernel. $0 \leq dn < n$.																		
<i>scale</i>	The scaling factor for kernel to convert the input integer coefficients into floating-point coefficients: floating-point coefficient = integer coefficient * 2**(-scale)																		
<i>colormap</i>	Internal data structure for image dithering.																		
RETURN VALUES	The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.																		

mllib_ImageColorErrorDiffusionMxN(3MLIB)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageColorDitherInit(3MLIB)`, `mllib_ImageColorDitherFree(3MLIB)`,
`mllib_ImageColorErrorDiffusion3x3(3MLIB)`,
`mllib_ImageColorOrderedDither8x8(3MLIB)`,
`mllib_ImageColorOrderedDitherMxN(3MLIB)`, `attributes(5)`

mllib_ImageColorHSL2RGB(3MLIB)

NAME	mllib_ImageColorHSL2RGB – HSL to RGB color conversion
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageColorHSL2RGB(mllib_image *dst, const mllib_image *src);</pre>
DESCRIPTION	<p>The <code>mllib_ImageColorHSL2RGB()</code> function performs a conversion from hue/saturation/lightness to red/green/blue. The source and destination images must be three-channel images.</p> <p>It uses the following equations:</p> $\begin{aligned} L' &= L && \text{if } L \leq 1/2 \\ L' &= 1 - L && \text{if } L > 1/2 \end{aligned}$ $\begin{aligned} V &= L + S \cdot L' \\ P &= L - S \cdot L' \\ Q &= L + S \cdot L' \cdot (1 - 2 \cdot \text{fraction}(H \cdot 6)) \\ T &= L - S \cdot L' \cdot (1 - 2 \cdot \text{fraction}(H \cdot 6)) \end{aligned}$ $\begin{aligned} R, G, B &= V, T, P && \text{if } 0 \leq H < 1/6 \\ R, G, B &= Q, V, P && \text{if } 1/6 \leq H < 2/6 \\ R, G, B &= P, V, T && \text{if } 2/6 \leq H < 3/6 \\ R, G, B &= P, Q, V && \text{if } 3/6 \leq H < 4/6 \\ R, G, B &= T, P, V && \text{if } 4/6 \leq H < 5/6 \\ R, G, B &= V, P, Q && \text{if } 5/6 \leq H < 1 \end{aligned}$ <p>where $0 \leq H < 1$ and $0 \leq S, L, L', V, P, Q, T, R, G, B \leq 1$.</p> <p>Assuming a pixel in the source image is (h, s, l) and its corresponding pixel in the destination image is (r, g, b), then for <code>MLIB_BYTE</code> images, the following applies:</p> $\begin{aligned} H &= h/256 \\ S &= s/255 \\ L &= l/255 \\ r &= R \cdot 255 \\ g &= G \cdot 255 \\ b &= B \cdot 255 \end{aligned}$ <p>for <code>MLIB_SHORT</code> images, the following applies:</p> $\begin{aligned} H &= (h + 32768) / 65536 \\ S &= (s + 32768) / 65535 \\ L &= (l + 32768) / 65535 \\ r &= R \cdot 65535 - 32768 \\ g &= G \cdot 65535 - 32768 \\ b &= B \cdot 65535 - 32768 \end{aligned}$ <p>for <code>MLIB_USHORT</code> images, the following applies:</p> $\begin{aligned} H &= h/65536 \\ S &= s/65535 \\ L &= l/65535 \\ r &= R \cdot 65535 \\ g &= G \cdot 65535 \\ b &= B \cdot 65535 \end{aligned}$

and for MLIB_INT images, the following applies:

```
H = (h + 2147483648) / 4294967296
S = (s + 2147483648) / 4294967295
L = (l + 2147483648) / 4294967295
r = R*4294967295 - 2147483648
g = G*4294967295 - 2147483648
b = B*4294967295 - 2147483648
```

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.
src Pointer to source image.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_ImageColorHSL2RGB_Fp\(3MLIB\)](#), [mllib_ImageColorRGB2HSL\(3MLIB\)](#), [mllib_ImageColorRGB2HSL_Fp\(3MLIB\)](#), [attributes\(5\)](#)

mllib_ImageColorHSL2RGB_Fp(3MLIB)

NAME	mllib_ImageColorHSL2RGB_Fp – HSL to RGB color conversion						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageColorHSL2RGB_Fp(mllib_image *dst, const mllib_image *src);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageColorHSL2RGB_Fp()</code> function performs a conversion from hue/saturation/lightness to red/green/blue. The source and destination images must be three-channel images.</p> <p>It uses the following equations:</p> $\begin{aligned} L' &= L && \text{if } L \leq 1/2 \\ L' &= 1 - L && \text{if } L > 1/2 \end{aligned}$ $\begin{aligned} V &= L + S \cdot L' \\ P &= L - S \cdot L' \\ Q &= L + S \cdot L' \cdot (1 - 2 \cdot \text{fraction}(H \cdot 6)) \\ T &= L - S \cdot L' \cdot (1 - 2 \cdot \text{fraction}(H \cdot 6)) \end{aligned}$ $\begin{aligned} R, G, B &= V, T, P && \text{if } 0 \leq H < 1/6 \\ R, G, B &= Q, V, P && \text{if } 1/6 \leq H < 2/6 \\ R, G, B &= P, V, T && \text{if } 2/6 \leq H < 3/6 \\ R, G, B &= P, Q, V && \text{if } 3/6 \leq H < 4/6 \\ R, G, B &= T, P, V && \text{if } 4/6 \leq H < 5/6 \\ R, G, B &= V, P, Q && \text{if } 5/6 \leq H < 1 \end{aligned}$ <p>where $0 \leq H < 1$ and $0 \leq S, L, L', V, P, Q, T, R, G, B \leq 1$.</p> <p>For <code>MLIB_FLOAT</code> and <code>MLIB_DOUBLE</code> images, the above equations are followed verbatim. Input <i>H</i> component values must be limited to the $[0.0, 1.0)$ range. Input <i>S</i> and <i>L</i> component values must be limited to the $[0.0, 1.0]$ range.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_ImageColorHSL2RGB(3MLIB) , mllib_ImageColorRGB2HSL(3MLIB) , mllib_ImageColorRGB2HSL_Fp(3MLIB) , attributes(5)						

NAME	mllib_ImageColorHSV2RGB – HSV to RGB color conversion
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageColorHSV2RGB(mllib_image *dst, const mllib_image *src);</pre>
DESCRIPTION	<p>The <code>mllib_ImageColorHSV2RGB()</code> function performs a conversion from hue/saturation/value to red/green/blue. The source and destination images must be three-channel images.</p> <p>It uses the following equations:</p> $P = V * (1 - S)$ $Q = V * (1 - S * \text{fraction}(H * 6))$ $T = V * (1 - S * (1 - \text{fraction}(H * 6)))$ <p>R, G, B = V, T, P if $0 \leq H < 1/6$ R, G, B = Q, V, P if $1/6 \leq H < 2/6$ R, G, B = P, V, T if $2/6 \leq H < 3/6$ R, G, B = P, Q, V if $3/6 \leq H < 4/6$ R, G, B = T, P, V if $4/6 \leq H < 5/6$ R, G, B = V, P, Q if $5/6 \leq H < 1$</p> <p>where $0 \leq H < 1$ and $0 \leq S, V, P, Q, T, R, G, B \leq 1$.</p> <p>Assuming a pixel in the source image is (h, s, v) and its corresponding pixel in the destination image is (r, g, b), then for <code>MLIB_BYTE</code> images, the following applies:</p> $H = h / 256$ $S = s / 255$ $V = v / 255$ $r = R * 255$ $g = G * 255$ $b = B * 255$ <p>for <code>MLIB_SHORT</code> images, the following applies:</p> $H = (h + 32768) / 65536$ $S = (s + 32768) / 65535$ $V = (v + 32768) / 65535$ $r = R * 65535 - 32768$ $g = G * 65535 - 32768$ $b = B * 65535 - 32768$ <p>for <code>MLIB_USHORT</code> images, the following applies:</p> $H = h / 65536$ $S = s / 65535$ $V = v / 65535$ $r = R * 65535$ $g = G * 65535$ $b = B * 65535$ <p>and for <code>MLIB_INT</code> images, the following applies:</p> $H = (h + 2147483648) / 4294967296$ $S = (s + 2147483648) / 4294967295$

mllib_ImageColorHSV2RGB(3MLIB)

```
V = (v + 2147483648) / 4294967295
r = R*4294967295 - 2147483648
g = G*4294967295 - 2147483648
b = B*4294967295 - 2147483648
```

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_ImageColorHSV2RGB_Fp\(3MLIB\)](#), [mllib_ImageColorRGB2HSV\(3MLIB\)](#), [mllib_ImageColorRGB2HSV_Fp\(3MLIB\)](#), [attributes\(5\)](#)

NAME mllib_ImageColorHSV2RGB_Fp – HSV to RGB color conversion

SYNOPSIS cc [*flag...*] *file...* -lmllib [*library...*]
 #include <mllib.h>

```
mllib_status mllib_ImageColorHSV2RGB_Fp(mllib_image *dst, const
    mllib_image *src);
```

DESCRIPTION The mllib_ImageColorHSV2RGB_Fp() function performs a conversion from hue/saturation/value to red/green/blue. The source and destination images must be three-channel images.

It uses the following equations:

$$\begin{aligned}
 P &= V*(1 - S) \\
 Q &= V*(1 - S*\text{fraction}(H*6)) \\
 T &= V*(1 - S*(1 - \text{fraction}(H*6)))
 \end{aligned}$$

$R, G, B = V, T, P$ if $0 \leq H < 1/6$
 $R, G, B = Q, V, P$ if $1/6 \leq H < 2/6$
 $R, G, B = P, V, T$ if $2/6 \leq H < 3/6$
 $R, G, B = P, Q, V$ if $3/6 \leq H < 4/6$
 $R, G, B = T, P, V$ if $4/6 \leq H < 5/6$
 $R, G, B = V, P, Q$ if $5/6 \leq H < 1$

where $0 \leq H < 1$ and $0 \leq S, V, P, Q, T, R, G, B \leq 1$.

For MLIB_FLOAT and MLIB_DOUBLE images, the above equations are followed verbatim. Input H component values must be limited to the [0.0, 1.0) range. Input S and V component values must be limited to the [0.0, 1.0] range.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.
src Pointer to source image.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageColorHSV2RGB(3MLIB), mllib_ImageColorRGB2HSV(3MLIB), mllib_ImageColorRGB2HSV_Fp(3MLIB), attributes(5)

mllib_ImageColorOrderedDither8x8(3MLIB)

NAME	mllib_ImageColorOrderedDither8x8 – true color to indexed color conversion using ordered dithering										
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status (mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, const mllib_s32 *<i>dmask</i>, mllib_s32 <i>scale</i>, const void *<i>colormap</i>);</pre>										
DESCRIPTION	<p>The <code>mllib_ImageColorOrderedDither8x8()</code> function converts a true color image to a pseudo color image with the method of ordered dithering. The source image can be an <code>MLIB_BYTE</code> or <code>MLIB_SHORT</code> image with three or four channels. The destination must be a single-channel <code>MLIB_BYTE</code> or <code>MLIB_SHORT</code> image.</p> <p>This function works only with a colorcube, rather than a general lookup table. The last parameter, <code>colormap</code>, is an internal data structure (which may include a colorcube) for inverse color mapping. Create it by calling the <code>mllib_ImageColorTrue2IndexInit()</code> function.</p>										
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>dst</i></td><td>Pointer to destination or destination image.</td></tr><tr><td><i>src</i></td><td>Pointer to source or source image.</td></tr><tr><td><i>dmask</i></td><td>Pointer to the 8x8 dither mask, in row major order. The dither mask is transposed differently for different channels to reduce artifacts.</td></tr><tr><td><i>scale</i></td><td>Scaling factor for <i>dmask</i> to convert the input integer coefficients into floating-point coefficients: floating-point coefficient = integer coefficient * 2**(-scale)</td></tr><tr><td><i>colormap</i></td><td>Internal data structure for inverse color mapping.</td></tr></table>	<i>dst</i>	Pointer to destination or destination image.	<i>src</i>	Pointer to source or source image.	<i>dmask</i>	Pointer to the 8x8 dither mask, in row major order. The dither mask is transposed differently for different channels to reduce artifacts.	<i>scale</i>	Scaling factor for <i>dmask</i> to convert the input integer coefficients into floating-point coefficients: floating-point coefficient = integer coefficient * 2**(-scale)	<i>colormap</i>	Internal data structure for inverse color mapping.
<i>dst</i>	Pointer to destination or destination image.										
<i>src</i>	Pointer to source or source image.										
<i>dmask</i>	Pointer to the 8x8 dither mask, in row major order. The dither mask is transposed differently for different channels to reduce artifacts.										
<i>scale</i>	Scaling factor for <i>dmask</i> to convert the input integer coefficients into floating-point coefficients: floating-point coefficient = integer coefficient * 2**(-scale)										
<i>colormap</i>	Internal data structure for inverse color mapping.										
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .										
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:										
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe				
ATTRIBUTE TYPE	ATTRIBUTE VALUE										
Interface Stability	Evolving										
MT-Level	MT-Safe										
SEE ALSO	<code>mllib_ImageColorErrorDiffusion3x3(3MLIB)</code> , <code>mllib_ImageColorTrue2Index(3MLIB)</code> , <code>mllib_ImageColorTrue2IndexFree(3MLIB)</code> , <code>mllib_ImageColorTrue2IndexInit(3MLIB)</code> , <code>attributes(5)</code>										

mllib_ImageColorOrderedDitherMxN(3MLIB)

NAME mllib_ImageColorOrderedDitherMxN – true-color to indexed-color or grayscale to black-white conversion, using ordered dithering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageColorOrderedDitherMxN(mllib_image *dst,
        const mllib_image *src, const mllib_s32 **dmask, mllib_s32 m,
        mllib_s32 n, mllib_s32 scale, const void *colormap);
```

DESCRIPTION The `mllib_ImageColorOrderedDitherMxN()` function converts a 3-channel image to a 1-channel indexed image, or converts a 1-channel grayscale image to a 1-channel MLIB_BIT image, with the method of ordered dithering.

The `src` can be an MLIB_BYTE image with 1 or 3 channels. The `dst` must be a 1-channel MLIB_BIT or MLIB_BYTE image.

The `colormap` must be created by `mllib_ImageColorDitherInit()`, and it must have a colorcube included.

The dither masks are required to have the following property:

$$0 \leq dmask[i][j] < 2^{**scale}; \text{ scale} > 0$$

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

dmask Pointer to the dither masks, one per channel, in row major order.

m Mask width. $m > 1$.

n Mask height. $n > 1$.

scale Scaling factor for *dmask* to convert the input integer coefficients into floating-point coefficients:

$$\text{floating-point coefficient} = \text{integer coefficient} * 2^{*(-scale)}$$

colormap Internal data structure for image dithering.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

`mllib_ImageColorOrderedDitherMxN(3MLIB)`

SEE ALSO | `mllib_ImageColorDitherInit(3MLIB)`, `mllib_ImageColorDitherFree(3MLIB)`,
`mllib_ImageColorErrorDiffusion3x3(3MLIB)`,
`mllib_ImageColorErrorDiffusionMxN(3MLIB)`,
`mllib_ImageColorOrderedDither8x8(3MLIB)`, `attributes(5)`

mlib_ImageColorRGB2CIEMono(3MLIB)

NAME mlib_ImageColorRGB2CIEMono – RGB to monochrome conversion

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_ImageColorRGB2CIEMono(mlib_image *dst, const
mlib_image *src);
```

DESCRIPTION The `mlib_ImageColorRGB2CIEMono()` function performs a conversion from a red/green/blue to a monochromatic image. The source image must be a three-channel image. The destination image must be a single-channel image.

It uses the following equation:

$$\text{dst}[x][y][0] = 0.2125 * \text{src}[x][y][0] + 0.7154 * \text{src}[x][y][1] + 0.0721 * \text{src}[x][y][2]$$

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mlib_ImageColorRGB2CIEMono_Fp(3MLIB)`, `mlib_ImageColorRGB2Mono(3MLIB)`, `mlib_ImageColorRGB2Mono_Fp(3MLIB)`, `attributes(5)`

mllib_ImageColorRGB2CIEMono_Fp(3MLIB)

NAME	mllib_ImageColorRGB2CIEMono_Fp – RGB to monochrome conversion						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageColorRGB2CIEMono_Fp(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageColorRGB2CIEMono_Fp()</code> function performs a conversion from a red/green/blue to a monochromatic image. The source image must be a three-channel image. The destination image must be a single-channel image.</p> <p>It uses the following equation:</p> $\text{dst}[x][y][0] = 0.2125 * \text{src}[x][y][0] + 0.7154 * \text{src}[x][y][1] + 0.0721 * \text{src}[x][y][2]$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageColorRGB2CIEMono(3MLIB)</code> , <code>mllib_ImageColorRGB2Mono(3MLIB)</code> , <code>mllib_ImageColorRGB2Mono_Fp(3MLIB)</code> , <code>attributes(5)</code>						

NAME	mllib_ImageColorRGB2HSL – RGB to HSL color conversion
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageColorRGB2HSL(mllib_image *dst, const mllib_image *src);</pre>
DESCRIPTION	<p>The <code>mllib_ImageColorRGB2HSL()</code> function performs a conversion from red/green/blue to hue/saturation/lightness. The source and destination images must be three-channel images.</p> <p>It uses the following equations:</p> <pre>V = max(R, G, B) Vmin = min(R, G, B) L = (V + Vmin)/2 S = (V - Vmin)/(V + Vmin) if L ≤ 1/2 S = (V - Vmin)/(2 - V - Vmin) if L > 1/2 H = (5.0 + (V - B)/(V - Vmin))/6 if R = V and G = Vmin H = (1.0 - (V - G)/(V - Vmin))/6 if R = V and B = Vmin H = (1.0 + (V - R)/(V - Vmin))/6 if G = V and B = Vmin H = (3.0 - (V - B)/(V - Vmin))/6 if G = V and R = Vmin H = (3.0 + (V - G)/(V - Vmin))/6 if B = V and R = Vmin H = (5.0 - (V - R)/(V - Vmin))/6 if B = V and G = Vmin H = 0.0 if R = G = B</pre> <p>where $0 \leq R, G, B, V, Vmin, L, S \leq 1$ and $0 \leq H < 1$.</p> <p>Assuming a pixel in the source image is (r, g, b) and its corresponding pixel in the destination image is (h, s, l), then for <code>MLIB_BYTE</code> images, the following applies:</p> <pre>R = r/255 G = g/255 B = b/255 h = H*256 s = S*255 l = L*255</pre> <p>for <code>MLIB_SHORT</code> images, the following applies:</p> <pre>R = (r + 32768)/65535 G = (g + 32768)/65535 B = (b + 32768)/65535 h = H*65536 - 32768 s = S*65535 - 32768 l = L*65535 - 32768</pre> <p>for <code>MLIB_USHORT</code> images, the following applies:</p> <pre>R = r/65535 G = g/65535 B = b/65535 h = H*65536 s = S*65535 l = L*65535</pre>

mllib_ImageColorRGB2HSL(3MLIB)

and for `MLIB_INT` images, the following applies:

```
R = (r + 2147483648) / 4294967295
G = (g + 2147483648) / 4294967295
B = (b + 2147483648) / 4294967295
h = H*4294967296 - 2147483648
s = S*4294967295 - 2147483648
l = L*4294967295 - 2147483648
```

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageColorHSL2RGB(3MLIB)`, `mllib_ImageColorHSL2RGB_Fp(3MLIB)`, `mllib_ImageColorRGB2HSL_Fp(3MLIB)`, `attributes(5)`

NAME mllib_ImageColorRGB2HSL_Fp – RGB to HSL color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageColorRGB2HSL_Fp(mllib_image *dst, const
    mllib_image *src);
```

DESCRIPTION The `mllib_ImageColorRGB2HSL_Fp()` function performs a conversion from red/green/blue to hue/saturation/lightness. The source and destination images must be three-channel images.

It uses the following equations:

$$V = \max(R, G, B)$$

$$Vmin = \min(R, G, B)$$

$$L = (V + Vmin) / 2$$

$$S = (V - Vmin) / (V + Vmin) \quad \text{if } L \leq 1/2$$

$$S = (V - Vmin) / (2 - V - Vmin) \quad \text{if } L > 1/2$$

$$H = (5.0 + (V - B) / (V - Vmin)) / 6 \quad \text{if } R = V \text{ and } G = Vmin$$

$$H = (1.0 - (V - G) / (V - Vmin)) / 6 \quad \text{if } R = V \text{ and } B = Vmin$$

$$H = (1.0 + (V - R) / (V - Vmin)) / 6 \quad \text{if } G = V \text{ and } B = Vmin$$

$$H = (3.0 - (V - B) / (V - Vmin)) / 6 \quad \text{if } G = V \text{ and } R = Vmin$$

$$H = (3.0 + (V - G) / (V - Vmin)) / 6 \quad \text{if } B = V \text{ and } R = Vmin$$

$$H = (5.0 - (V - R) / (V - Vmin)) / 6 \quad \text{if } B = V \text{ and } G = Vmin$$

$$H = 0.0 \quad \text{if } R = G = B$$

where $0 \leq R, G, B, V, Vmin, L, S \leq 1$ and $0 \leq H < 1$.

For `MLIB_FLOAT` and `MLIB_DOUBLE` images, the above equations are followed verbatim. Input R, G, and B component values must be limited to the [0.0, 1.0] range.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageColorHSL2RGB(3MLIB)`, `mllib_ImageColorHSL2RGB_Fp(3MLIB)`, `mllib_ImageColorRGB2HSL(3MLIB)`, `attributes(5)`

mllib_ImageColorRGB2HSV(3MLIB)

NAME	mllib_ImageColorRGB2HSV – RGB to HSV color conversion
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageColorRGB2HSV(mllib_image *dst, const mllib_image *src);</pre>
DESCRIPTION	<p>The <code>mllib_ImageColorRGB2HSV()</code> function performs a conversion from red/green/blue to hue/saturation/value. The source and destination images must be three-channel images.</p> <p>It uses the following equations:</p> $V = \max(R, G, B)$ $V_{\min} = \min(R, G, B)$ $S = (V - V_{\min})/V$ $H = (5.0 + (V - B)/(V - V_{\min}))/6 \quad \text{if } R = V \text{ and } G = V_{\min}$ $H = (1.0 - (V - G)/(V - V_{\min}))/6 \quad \text{if } R = V \text{ and } B = V_{\min}$ $H = (1.0 + (V - R)/(V - V_{\min}))/6 \quad \text{if } G = V \text{ and } B = V_{\min}$ $H = (3.0 - (V - B)/(V - V_{\min}))/6 \quad \text{if } G = V \text{ and } R = V_{\min}$ $H = (3.0 + (V - G)/(V - V_{\min}))/6 \quad \text{if } B = V \text{ and } R = V_{\min}$ $H = (5.0 - (V - R)/(V - V_{\min}))/6 \quad \text{if } B = V \text{ and } G = V_{\min}$ $H = 0.0 \quad \text{if } R = G = B$ <p>where $0 \leq R, G, B, V, V_{\min}, S \leq 1$ and $0 \leq H < 1$.</p> <p>Assuming a pixel in the source image is (r, g, b) and its corresponding pixel in the destination image is (h, s, v), then for <code>MLIB_BYTE</code> images, the following applies:</p> $R = r/255$ $G = g/255$ $B = b/255$ $h = H*256$ $s = S*255$ $v = V*255$ <p>for <code>MLIB_SHORT</code> images, the following applies:</p> $R = (r + 32768)/65535$ $G = (g + 32768)/65535$ $B = (b + 32768)/65535$ $h = H*65536 - 32768$ $s = S*65535 - 32768$ $v = V*65535 - 32768$ <p>for <code>MLIB_USHORT</code> images, the following applies:</p> $R = r/65535$ $G = g/65535$ $B = b/65535$ $h = H*65536$ $s = S*65535$ $v = V*65535$ <p>and for <code>MLIB_INT</code> images, the following applies:</p>

mllib_ImageColorRGB2HSV(3MLIB)

```
R = (r + 2147483648) / 4294967295
G = (g + 2147483648) / 4294967295
B = (b + 2147483648) / 4294967295
h = H*4294967296 - 2147483648
s = S*4294967295 - 2147483648
v = V*4294967295 - 2147483648
```

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.
src Pointer to source image.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageColorHSV2RGB(3MLIB)`, `mllib_ImageColorHSV2RGB_Fp(3MLIB)`, `mllib_ImageColorRGB2HSV_Fp(3MLIB)`, `attributes(5)`

mllib_ImageColorRGB2HSV_Fp(3MLIB)

NAME	mllib_ImageColorRGB2HSV_Fp – RGB to HSV color conversion						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageColorRGB2HSV_Fp(mllib_image *dst, const mllib_image *src);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageColorRGB2HSV_Fp()</code> function performs a conversion from red/green/blue to hue/saturation/value. The source and destination images must be three-channel images.</p> <p>It uses the following equations:</p> $V = \max(R, G, B)$ $V_{\min} = \min(R, G, B)$ $S = (V - V_{\min})/V$ $H = (5.0 + (V - B)/(V - V_{\min}))/6 \quad \text{if } R = V \text{ and } G = V_{\min}$ $H = (1.0 - (V - G)/(V - V_{\min}))/6 \quad \text{if } R = V \text{ and } B = V_{\min}$ $H = (1.0 + (V - R)/(V - V_{\min}))/6 \quad \text{if } G = V \text{ and } B = V_{\min}$ $H = (3.0 - (V - B)/(V - V_{\min}))/6 \quad \text{if } G = V \text{ and } R = V_{\min}$ $H = (3.0 + (V - G)/(V - V_{\min}))/6 \quad \text{if } B = V \text{ and } R = V_{\min}$ $H = (5.0 - (V - R)/(V - V_{\min}))/6 \quad \text{if } B = V \text{ and } G = V_{\min}$ $H = 0.0 \quad \text{if } R = G = B$ <p>where $0 \leq R, G, B, V, V_{\min}, S \leq 1$ and $0 \leq H < 1$.</p> <p>For <code>MLIB_FLOAT</code> and <code>MLIB_DOUBLE</code> images, the above equations are followed verbatim. Input <code>R</code>, <code>G</code>, and <code>B</code> component values must be limited to the <code>[0.0, 1.0]</code> range.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageColorHSV2RGB(3MLIB)</code> , <code>mllib_ImageColorHSV2RGB_Fp(3MLIB)</code> , <code>mllib_ImageColorRGB2HSV(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageColorRGB2Mono – RGB to monochrome conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageColorRGB2Mono(mllib_image *dst, const
    mllib_image *src, const mllib_d64 *weight);
```

DESCRIPTION The `mllib_ImageColorRGB2Mono()` function performs a conversion from a red/green/blue to a monochromatic image. The source image must be a three-channel image. The destination image must be a single-channel image.

It uses the following equation:

$$dst[x][y][0] = weight[0]*src[x][y][0] + weight[1]*src[x][y][1] + weight[2]*src[x][y][2]$$

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

weight Array of three blending coefficients. It is recommended that these sum to 1.0, but it is not required.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageColorRGB2CIEMono(3MLIB)`,
`mllib_ImageColorRGB2CIEMono_Fp(3MLIB)`,
`mllib_ImageColorRGB2Mono_Fp(3MLIB)`, `attributes(5)`

mllib_ImageColorRGB2Mono_Fp(3MLIB)

NAME	mllib_ImageColorRGB2Mono_Fp – RGB to monochrome conversion						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageColorRGB2Mono_Fp(mllib_image *dst, const mllib_image *src, const mllib_d64 *weight);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageColorRGB2Mono_Fp()</code> function performs a conversion from a red/green/blue to a monochromatic image. The source image must be a three-channel image. The destination image must be a single-channel image.</p> <p>It uses the following equation:</p> $\text{dst}[x][y][0] = \text{weight}[0]*\text{src}[x][y][0] + \text{weight}[1]*\text{src}[x][y][1] + \text{weight}[2]*\text{src}[x][y][2]$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>weight</i> Array of three blending coefficients. It is recommended that these sum to 1.0, but it is not required.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageColorRGB2CIEMono(3MLIB)</code> , <code>mllib_ImageColorRGB2CIEMono_Fp(3MLIB)</code> , <code>mllib_ImageColorRGB2Mono(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageColorRGB2XYZ – RGB to XYZ color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageColorRGB2XYZ(mllib_image *dst, const
mllib_image *src);
```

DESCRIPTION The mllib_ImageColorRGB2XYZ() function performs a color space conversion from ITU-R Rec.708 RGB with D64 white point to CIE XYZ.

The source and destination images must be three-channel images.

It uses the following equation:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \text{cmat}[0] & \text{cmat}[1] & \text{cmat}[2] \\ \text{cmat}[3] & \text{cmat}[4] & \text{cmat}[5] \\ \text{cmat}[6] & \text{cmat}[7] & \text{cmat}[8] \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

where

```
cmat[] = { 0.412453, 0.357580, 0.180423,
           0.212671, 0.715160, 0.072169,
           0.019334, 0.119193, 0.950227 };
src[x][y] = { R, G, B };
dst[x][y] = { X, Y, Z };
```

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

RETURN VALUES The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageColorConvert1(3MLIB), mllib_ImageColorConvert1_Fp(3MLIB), mllib_ImageColorRGB2XYZ_Fp(3MLIB), mllib_ImageColorXYZ2RGB(3MLIB), mllib_ImageColorXYZ2RGB_Fp(3MLIB), mllib_ImageColorYCC2RGB(3MLIB), mllib_ImageColorYCC2RGB_Fp(3MLIB), attributes(5)

mllib_ImageColorRGB2XYZ_Fp(3MLIB)

NAME	mllib_ImageColorRGB2XYZ_Fp – RGB to XYZ color conversion						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageColorRGB2XYZ_Fp(mllib_image *dst, const mllib_image *src);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageColorRGB2XYZ_Fp()</code> function performs a color space conversion from ITU-R Rec.708 RGB with D64 white point to CIE XYZ.</p> <p>The source and destination images must be three-channel images.</p> <p>It uses the following equation:</p> $\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \text{cmat}[0] & \text{cmat}[1] & \text{cmat}[2] \\ \text{cmat}[3] & \text{cmat}[4] & \text{cmat}[5] \\ \text{cmat}[6] & \text{cmat}[7] & \text{cmat}[8] \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix}$ <p>where</p> <pre>cmat[] = { 0.412453, 0.357580, 0.180423, 0.212671, 0.715160, 0.072169, 0.019334, 0.119193, 0.950227 }; src[x][y] = { R, G, B }; dst[x][y] = { X, Y, Z };</pre>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageColorConvert1(3MLIB)</code> , <code>mllib_ImageColorConvert1_Fp(3MLIB)</code> , <code>mllib_ImageColorRGB2XYZ(3MLIB)</code> , <code>mllib_ImageColorXYZ2RGB(3MLIB)</code> , <code>mllib_ImageColorXYZ2RGB_Fp(3MLIB)</code> , <code>mllib_ImageColorYCC2RGB(3MLIB)</code> , <code>mllib_ImageColorYCC2RGB_Fp(3MLIB)</code> , <code>mllib_ImageColorYCC2RGB_Fp(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageColorRGB2YCC – RGB to YCC color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageColorRGB2YCC(mllib_image *dst, const
mllib_image *src);
```

DESCRIPTION The mllib_ImageColorRGB2YCC() function performs a color space conversion from computer R'G'B' to ITU-R Rec.601 Y'CbCr.

The source and destination images must be three-channel images.

It uses the following equation:

$$\begin{bmatrix} Y' \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} cmat[0] & cmat[1] & cmat[2] \\ cmat[3] & cmat[4] & cmat[5] \\ cmat[6] & cmat[7] & cmat[8] \end{bmatrix} * \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} + \begin{bmatrix} offset[0] \\ offset[1] \\ offset[2] \end{bmatrix}$$

where

```
cmat[] = { 65.738/256, 129.057/256, 25.064/256,
-37.945/256, -74.494/256, 112.439/256,
112.439/256, -94.154/256, -18.285/256 };
offset[] = { 16, 128, 128 };
src[x][y] = { R', G', B' };
dst[x][y] = { Y', Cb, Cr };
```

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageColorConvert2(3MLIB), mllib_ImageColorConvert2_Fp(3MLIB), mllib_ImageColorRGB2YCC_Fp(3MLIB), mllib_ImageColorXYZ2RGB(3MLIB), mllib_ImageColorXYZ2RGB_Fp(3MLIB), mllib_ImageColorYCC2RGB(3MLIB), mllib_ImageColorYCC2RGB_Fp(3MLIB), attributes(5)

mllib_ImageColorRGB2YCC_Fp(3MLIB)

NAME	mllib_ImageColorRGB2YCC_Fp – RGB to YCC color conversion						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageColorRGB2YCC_Fp(mllib_image *dst, const mllib_image *src);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageColorRGB2YCC_Fp()</code> function performs a color space conversion from computer R'G'B' to ITU-R Rec.601 Y'CbCr.</p> <p>The source and destination images must be three-channel images.</p> <p>It uses the following equation:</p> $\begin{array}{l} Y' \quad cmat[0] \ cmat[1] \ cmat[2] \quad R' \quad offset[0] \\ Cb = cmat[3] \ cmat[4] \ cmat[5] * G' + offset[1] \\ Cr \quad cmat[6] \ cmat[7] \ cmat[8] \quad B' \quad offset[2] \end{array}$ <p>where</p> <pre>cmat[] = { 65.738/256, 129.057/256, 25.064/256, -37.945/256, -74.494/256, 112.439/256, 112.439/256, -94.154/256, -18.285/256 }; offset[] = { 16, 128, 128 }; src[x][y] = { R', G', B' }; dst[x][y] = { Y', Cb, Cr };</pre>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageColorConvert2(3MLIB)</code> , <code>mllib_ImageColorConvert2_Fp(3MLIB)</code> , <code>mllib_ImageColorRGB2YCC(3MLIB)</code> , <code>mllib_ImageColorXYZ2RGB(3MLIB)</code> , <code>mllib_ImageColorXYZ2RGB_Fp(3MLIB)</code> , <code>mllib_ImageColorYCC2RGB(3MLIB)</code> , <code>mllib_ImageColorYCC2RGB_Fp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageColorTrue2Index(3MLIB)

NAME mllib_ImageColorTrue2Index – true color to indexed color using nearest matched LUT entries

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageColorTrue2Index(mllib_image *dst, const
    mllib_image *src, const void *colormap);
```

DESCRIPTION The `mllib_ImageColorTrue2Index()` function converts a true color image to a pseudo color image with the method of finding the nearest matched lookup table entry for each pixel. The source image can be an `MLIB_BYTE` or `MLIB_SHORT` image with three or four channels. The destination must be a single-channel `MLIB_BYTE` or `MLIB_SHORT` image.

The last parameter, *colormap*, is an internal data structure (which includes the lookup table) for inverse color mapping. Create it by calling the `mllib_ImageColorTrue2IndexInit()` function.

PARAMETERS The function takes the following arguments:

- dst* Pointer to destination or destination image.
- src* Pointer to source or source image.
- colormap* Internal data structure for inverse color mapping.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageColorErrorDiffusion3x3(3MLIB)`,
`mllib_ImageColorOrderedDither8x8(3MLIB)`,
`mllib_ImageColorTrue2IndexFree(3MLIB)`,
`mllib_ImageColorTrue2IndexInit(3MLIB)`, `attributes(5)`

mllib_ImageColorTrue2IndexFree(3MLIB)

NAME mllib_ImageColorTrue2IndexFree – releases the internal data structure for true color to indexed color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_ImageColorTrue2IndexFree(void *colormap);
```

DESCRIPTION The mllib_ImageColorTrue2IndexFree() function releases the internal data structure, *colormap*, which was created by mllib_ImageColorTrue2IndexInit() and was used by one of the following functions:

mllib_ImageColorTrue2Index
mllib_ImageColorErrorDiffusion3x3
mllib_ImageColorOrderedDither8x8

PARAMETERS The function takes the following arguments:

colormap Internal data structure for inverse color mapping.

RETURN VALUES None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageColorErrorDiffusion3x3(3MLIB),
mllib_ImageColorOrderedDither8x8(3MLIB),
mllib_ImageColorTrue2Index(3MLIB),
mllib_ImageColorTrue2IndexInit(3MLIB), attributes(5)

NAME	mllib_ImageColorTrue2IndexInit – initialization for true color to indexed color conversion																
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageColorTrue2IndexInit(void **<i>colormap</i>, mllib_s32 <i>bits</i>, mllib_type <i>intype</i>, mllib_type <i>outtype</i>, mllib_s32 <i>channels</i>, mllib_s32 <i>entries</i>, mllib_s32 <i>offset</i>, const void **<i>table</i>);</pre>																
DESCRIPTION	<p>The <code>mllib_ImageColorTrue2IndexInit()</code> function creates and initializes an internal data structure based on the input lookup table and other parameters for inverse color mapping.</p> <p>The lookup table can have either three or four channels. The number of channels of the lookup table should match that of the source image provided to the function that will use the <i>colormap</i> structure created by this function.</p> <p>The type of the lookup table can be one of the following:</p> <pre>MLIB_BYTE in, MLIB_BYTE out (i.e., BYTE-to-BYTE) MLIB_SHORT in, MLIB_SHORT out (i.e., SHORT-to-SHORT) MLIB_SHORT in, MLIB_BYTE out (i.e., SHORT-to-BYTE)</pre> <p>The input type of the lookup table should match the type of the destination image; the output type of the lookup table should match the source image type. The source and destination images are the images provided to the function that is going to use the <i>colormap</i> structure created by <code>mllib_ImageColorTrue2IndexInit()</code> to do inverse color mapping.</p>																
PARAMETERS	<p>The function takes the following arguments:</p> <table border="0"> <tr> <td style="padding-right: 20px;"><i>colormap</i></td> <td>Internal data structure for inverse color mapping.</td> </tr> <tr> <td><i>bits</i></td> <td>Number of bits per color component used in the colorcube of the colormap structure. (If <i>bits</i> = 0, then no colorcube is created. But the inverse color mapping might be done by using the original lookup table.)</td> </tr> <tr> <td><i>intype</i></td> <td>Data type of the source image and lookup table.</td> </tr> <tr> <td><i>outtype</i></td> <td>Data type of the destination indexed image.</td> </tr> <tr> <td><i>channels</i></td> <td>Number of channels of the lookup table.</td> </tr> <tr> <td><i>entries</i></td> <td>Number of entries of the lookup table.</td> </tr> <tr> <td><i>offset</i></td> <td>The first entry offset of the lookup table.</td> </tr> <tr> <td><i>table</i></td> <td>The lookup table (LUT).</td> </tr> </table>	<i>colormap</i>	Internal data structure for inverse color mapping.	<i>bits</i>	Number of bits per color component used in the colorcube of the colormap structure. (If <i>bits</i> = 0, then no colorcube is created. But the inverse color mapping might be done by using the original lookup table.)	<i>intype</i>	Data type of the source image and lookup table.	<i>outtype</i>	Data type of the destination indexed image.	<i>channels</i>	Number of channels of the lookup table.	<i>entries</i>	Number of entries of the lookup table.	<i>offset</i>	The first entry offset of the lookup table.	<i>table</i>	The lookup table (LUT).
<i>colormap</i>	Internal data structure for inverse color mapping.																
<i>bits</i>	Number of bits per color component used in the colorcube of the colormap structure. (If <i>bits</i> = 0, then no colorcube is created. But the inverse color mapping might be done by using the original lookup table.)																
<i>intype</i>	Data type of the source image and lookup table.																
<i>outtype</i>	Data type of the destination indexed image.																
<i>channels</i>	Number of channels of the lookup table.																
<i>entries</i>	Number of entries of the lookup table.																
<i>offset</i>	The first entry offset of the lookup table.																
<i>table</i>	The lookup table (LUT).																
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .																

mllib_ImageColorTrue2IndexInit(3MLIB)

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageColorErrorDiffusion3x3(3MLIB)`,
`mllib_ImageColorOrderedDither8x8(3MLIB)`,
`mllib_ImageColorTrue2Index(3MLIB)`,
`mllib_ImageColorTrue2IndexFree(3MLIB)`, `attributes(5)`

NAME mllib_ImageColorXYZ2RGB – XYZ to RGB color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageColorXYZ2RGB(mllib_image *dst, const
mllib_image *src);
```

DESCRIPTION The mllib_ImageColorXYZ2RGB() function performs a color space conversion from CIE XYZ to ITU-R Rec.708 RGB with D64 white point.

The source and destination images must be three-channel images.

It uses the following equation:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} \text{cmat}[0] & \text{cmat}[1] & \text{cmat}[2] \\ \text{cmat}[3] & \text{cmat}[4] & \text{cmat}[5] \\ \text{cmat}[6] & \text{cmat}[7] & \text{cmat}[8] \end{bmatrix} * \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

where

```
cmat[] = { 3.240479, -1.537150, -0.498535,
          -0.969256,  1.875992,  0.041566,
           0.055648, -0.204043,  1.057311 };
src[x][y] = { X, Y, Z };
dst[x][y] = { R, G, B };
```

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageColorConvert1(3MLIB), mllib_ImageColorConvert1_Fp(3MLIB), mllib_ImageColorRGB2XYZ(3MLIB), mllib_ImageColorRGB2XYZ_Fp(3MLIB), mllib_ImageColorRGB2YCC(3MLIB), mllib_ImageColorRGB2YCC_Fp(3MLIB), mllib_ImageColorXYZ2RGB_Fp(3MLIB), attributes(5)

mllib_ImageColorXYZ2RGB_Fp(3MLIB)

NAME	mllib_ImageColorXYZ2RGB_Fp – XYZ to RGB color conversion						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageColorXYZ2RGB_Fp(mllib_image *dst, const mllib_image *src);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageColorXYZ2RGB_Fp()</code> function performs a color space conversion from CIE XYZ to ITU-R Rec.708 RGB with D64 white point.</p> <p>The source and destination images must be three-channel images.</p> <p>It uses the following equation:</p> $\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} \text{cmat}[0] & \text{cmat}[1] & \text{cmat}[2] \\ \text{cmat}[3] & \text{cmat}[4] & \text{cmat}[5] \\ \text{cmat}[6] & \text{cmat}[7] & \text{cmat}[8] \end{bmatrix} * \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$ <p>where</p> <pre>cmat[] = { 3.240479, -1.537150, -0.498535, -0.969256, 1.875992, 0.041566, 0.055648, -0.204043, 1.057311 }; src[x][y] = { X, Y, Z }; dst[x][y] = { R, G, B };</pre>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageColorConvert1(3MLIB)</code> , <code>mllib_ImageColorConvert1_Fp(3MLIB)</code> , <code>mllib_ImageColorRGB2XYZ(3MLIB)</code> , <code>mllib_ImageColorRGB2XYZ_Fp(3MLIB)</code> , <code>mllib_ImageColorRGB2YCC(3MLIB)</code> , <code>mllib_ImageColorRGB2YCC_Fp(3MLIB)</code> , <code>mllib_ImageColorXYZ2RGB(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageColorYCC2RGB – YCC to RGB color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageColorYCC2RGB(mllib_image *dst, const
mllib_image *src);
```

DESCRIPTION The mllib_ImageColorYCC2RGB() function performs a color space conversion from ITU-R Rec.601 Y'CbCr to computer R'G'B'.

The source and destination images must be three-channel images.

It uses the following equation:

$$\begin{matrix} |R'| & | & |cmat[0] & cmat[1] & cmat[2]| & |Y'| & |offset[0]| \\ |G'| & = & |cmat[3] & cmat[4] & cmat[5]| & * & |Cb| & + & |offset[1]| \\ |B'| & | & |cmat[6] & cmat[7] & cmat[8]| & |Cr| & |offset[2]| \end{matrix}$$

where

```
cmat[] = { 298.082/256, 0.000/256, 408.583/256,
           298.082/256, -100.291/256, -208.120/256,
           298.082/256, 516.411/256, 0.000/256 };
offset[] = { -222.922, 135.575, -276.836 };
src[x][y] = { Y', Cb, Cr };
dst[x][y] = { R', G', B' };
```

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageColorConvert2(3MLIB), mllib_ImageColorConvert2_Fp(3MLIB), mllib_ImageColorRGB2XYZ(3MLIB), mllib_ImageColorRGB2XYZ_Fp(3MLIB), mllib_ImageColorRGB2YCC(3MLIB), mllib_ImageColorRGB2YCC_Fp(3MLIB), mllib_ImageColorYCC2RGB_Fp(3MLIB), attributes(5)

mllib_ImageColorYCC2RGB_Fp(3MLIB)

NAME	mllib_ImageColorYCC2RGB_Fp – YCC to RGB color conversion						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageColorYCC2RGB_Fp(mllib_image *dst, const mllib_image *src);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageColorYCC2RGB_Fp()</code> function performs a color space conversion from ITU-R Rec.601 Y'CbCr to computer R'G'B'.</p> <p>The source and destination images must be three-channel images.</p> <p>It uses the following equation:</p> $\begin{array}{l} R' \\ G' \\ B' \end{array} = \begin{array}{l} cmat[0] \ cmat[1] \ cmat[2] \\ cmat[3] \ cmat[4] \ cmat[5] \\ cmat[6] \ cmat[7] \ cmat[8] \end{array} * \begin{array}{l} Y' \\ Cb \\ Cr \end{array} + \begin{array}{l} offset[0] \\ offset[1] \\ offset[2] \end{array}$ <p>where</p> <pre>cmat[] = { 298.082/256, 0.000/256, 408.583/256, 298.082/256, -100.291/256, -208.120/256, 298.082/256, 516.411/256, 0.000/256 }; offset[] = { -222.922, 135.575, -276.836 }; src[x][y] = { Y', Cb, Cr }; dst[x][y] = { R', G', B' };</pre>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageColorConvert2(3MLIB)</code> , <code>mllib_ImageColorConvert2_Fp(3MLIB)</code> , <code>mllib_ImageColorRGB2XYZ(3MLIB)</code> , <code>mllib_ImageColorRGB2XYZ_Fp(3MLIB)</code> , <code>mllib_ImageColorRGB2YCC(3MLIB)</code> , <code>mllib_ImageColorRGB2YCC_Fp(3MLIB)</code> , <code>mllib_ImageColorYCC2RGB(3MLIB)</code> , <code>attributes(5)</code>						

NAME	mllib_ImageComposite – image composition																								
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageComposite(mllib_image *dst, const mllib_image *src1, const mllib_image *src2, mllib_blend bsrc1, mllib_blend bsrc2, mllib_s32 cmask);</pre>																								
DESCRIPTION	<p>The <code>mllib_ImageComposite()</code> function supports digital image composition.</p> <p>It is a wrapper of the <code>mllib_ImageBlend_BSCR1_BSRC2</code> group of functions and can perform various types of composition based on the parameters passed in, whereas each function in that group can perform only the one kind of composition denoted by its name.</p> <p>The image type must be <code>MLIB_BYTE</code>. The input and output images must contain three or four channels. For three-channel images, the alpha value is as if the alpha value is 1.</p> <p>The following are predefined blend factor types used in mediaLib image composition functions.</p> <pre>/* image blend factors */ typedef enum { MLIB_BLEND_ZERO, MLIB_BLEND_ONE, MLIB_BLEND_DST_COLOR, MLIB_BLEND_SRC_COLOR, MLIB_BLEND_ONE_MINUS_DST_COLOR, MLIB_BLEND_ONE_MINUS_SRC_COLOR, MLIB_BLEND_DST_ALPHA, MLIB_BLEND_SRC_ALPHA, MLIB_BLEND_ONE_MINUS_DST_ALPHA, MLIB_BLEND_ONE_MINUS_SRC_ALPHA, MLIB_BLEND_SRC_ALPHA_SATURATE } mllib_blend;</pre> <p>See the following table for the definitions of the blend factors.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Type</th> <th style="text-align: left;">Blend Factor [*]</th> <th style="text-align: left;">Abbr.</th> </tr> </thead> <tbody> <tr> <td><code>MLIB_BLEND_ZERO</code></td> <td>(0,0,0,0)</td> <td>ZERO</td> </tr> <tr> <td><code>MLIB_BLEND_ONE</code></td> <td>(1,1,1,1)</td> <td>ONE</td> </tr> <tr> <td><code>MLIB_BLEND_DST_COLOR</code></td> <td>(Rd,Gd,Bd,Ad)</td> <td>DC</td> </tr> <tr> <td><code>MLIB_BLEND_SRC_COLOR</code></td> <td>(Rs,Gs,Bs,As)</td> <td>SC</td> </tr> <tr> <td><code>MLIB_BLEND_ONE_MINUS_DST_COLOR</code></td> <td>(1,1,1,1)-(Rd,Gd,Bd,Ad)</td> <td>OMDC</td> </tr> <tr> <td><code>MLIB_BLEND_ONE_MINUS_SRC_COLOR</code></td> <td>(1,1,1,1)-(Rs,Gs,Bs,As)</td> <td>OMSC</td> </tr> <tr> <td><code>MLIB_BLEND_DST_ALPHA</code></td> <td>(Ad,Ad,Ad,Ad)</td> <td>DA</td> </tr> </tbody> </table>	Type	Blend Factor [*]	Abbr.	<code>MLIB_BLEND_ZERO</code>	(0,0,0,0)	ZERO	<code>MLIB_BLEND_ONE</code>	(1,1,1,1)	ONE	<code>MLIB_BLEND_DST_COLOR</code>	(Rd,Gd,Bd,Ad)	DC	<code>MLIB_BLEND_SRC_COLOR</code>	(Rs,Gs,Bs,As)	SC	<code>MLIB_BLEND_ONE_MINUS_DST_COLOR</code>	(1,1,1,1)-(Rd,Gd,Bd,Ad)	OMDC	<code>MLIB_BLEND_ONE_MINUS_SRC_COLOR</code>	(1,1,1,1)-(Rs,Gs,Bs,As)	OMSC	<code>MLIB_BLEND_DST_ALPHA</code>	(Ad,Ad,Ad,Ad)	DA
Type	Blend Factor [*]	Abbr.																							
<code>MLIB_BLEND_ZERO</code>	(0,0,0,0)	ZERO																							
<code>MLIB_BLEND_ONE</code>	(1,1,1,1)	ONE																							
<code>MLIB_BLEND_DST_COLOR</code>	(Rd,Gd,Bd,Ad)	DC																							
<code>MLIB_BLEND_SRC_COLOR</code>	(Rs,Gs,Bs,As)	SC																							
<code>MLIB_BLEND_ONE_MINUS_DST_COLOR</code>	(1,1,1,1)-(Rd,Gd,Bd,Ad)	OMDC																							
<code>MLIB_BLEND_ONE_MINUS_SRC_COLOR</code>	(1,1,1,1)-(Rs,Gs,Bs,As)	OMSC																							
<code>MLIB_BLEND_DST_ALPHA</code>	(Ad,Ad,Ad,Ad)	DA																							

mllib_ImageComposite(3MLIB)

Type	Blend Factor [*]	Abbr.
MLIB_BLEND_SRC_ALPHA	(As,As,As,As)	SA
MLIB_BLEND_ONE_MINUS_DST_ALPHA	(1,1,1,1)-(Ad,Ad,Ad,Ad)	OMDA
MLIB_BLEND_ONE_MINUS_SRC_ALPHA	(1,1,1,1)-(As,As,As,As)	OMSA
MLIB_BLEND_SRC_ALPHA_SATURATE	(f,f,f,1)	SAS

[*]: The components of the first source image pixel are (Rd, Gd, Bd, Ad), and the components of the second source pixel are (Rs, Gs, Bs, As). Function $f = \min(As, 1 - Ad)$.

The blending formula for non-in-place processing is:

$$Cd = Cs1 * S1 + Cs2 * S2$$

where Cd is the destination pixel (Rd, Gd, Bd, Ad), Cs1 is the first source pixel (Rs1, Gs1, Bs1, As1), Cs2 is the second source pixel (Rs2, Gs2, Bs2, As2), and S1 and S2 are the blend factors for the first and second sources, respectively.

PARAMETERS

The function takes the following arguments:

<i>dst</i>	Pointer to destination image.
<i>src1</i>	Pointer to the first source image.
<i>src2</i>	Pointer to the second source image.
<i>bsrc1</i>	Blend factor type for the first source image.
<i>bsrc2</i>	Blend factor type for the second source image.
<i>cmask</i>	Channel mask to indicate the alpha channel. Each bit of the mask represents a channel in the image. The channel corresponding to the 1 bit is the alpha channel. <i>cmask</i> must be either 0x01 or 0x08.

RETURN VALUES

The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES

See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

[mllib_ImageBlend_BSRC1_BSRC2\(3MLIB\)](#),
[mllib_ImageBlend_BSRC1_BSRC2_Inp\(3MLIB\)](#),
[mllib_ImageComposite_Inp\(3MLIB\)](#), [attributes\(5\)](#)

NAME mllib_ImageComposite_Inp – image composition, in place

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageComposite_Inp(mllib_image *src1dst, const
    mllib_image *src2, mllib_blend bsrc1, mllib_blend bsrc2, mllib_s32
    cmask);
```

DESCRIPTION

The `mllib_ImageComposite_Inp()` function supports digital image composition.

It is a wrapper of the `mllib_ImageBlend_BSCR1_BSRC2_Inp` group of functions and can perform various types of composition based on the parameters passed in, whereas each function in the `mllib_ImageBlend_BSCR1_BSRC2_Inp` group can perform only the one kind of composition denoted by its name.

The image type must be `MLIB_BYTE`. The input and output images must contain three or four channels. For three-channel images, the alpha value is as if the alpha value is 1.

The following are predefined blend factor types used in mediaLib image composition functions.

```
/* image blend factors */
typedef enum {
    MLIB_BLEND_ZERO,
    MLIB_BLEND_ONE,
    MLIB_BLEND_DST_COLOR,
    MLIB_BLEND_SRC_COLOR,
    MLIB_BLEND_ONE_MINUS_DST_COLOR,
    MLIB_BLEND_ONE_MINUS_SRC_COLOR,
    MLIB_BLEND_DST_ALPHA,
    MLIB_BLEND_SRC_ALPHA,
    MLIB_BLEND_ONE_MINUS_DST_ALPHA,
    MLIB_BLEND_ONE_MINUS_SRC_ALPHA,
    MLIB_BLEND_SRC_ALPHA_SATURATE
} mllib_blend;
```

See the following table for the definitions of the blend factors.

Type	Blend Factor [*]	Abbr.
<code>MLIB_BLEND_ZERO</code>	(0,0,0,0)	ZERO
<code>MLIB_BLEND_ONE</code>	(1,1,1,1)	ONE
<code>MLIB_BLEND_DST_COLOR</code>	(Rd,Gd,Bd,Ad)	DC
<code>MLIB_BLEND_SRC_COLOR</code>	(Rs,Gs,Bs,As)	SC
<code>MLIB_BLEND_ONE_MINUS_DST_COLOR</code>	(1,1,1,1)-(Rd,Gd,Bd,Ad)	OMDC
<code>MLIB_BLEND_ONE_MINUS_SRC_COLOR</code>	(1,1,1,1)-(Rs,Gs,Bs,As)	OMSC
<code>MLIB_BLEND_DST_ALPHA</code>	(Ad,Ad,Ad,Ad)	DA

mllib_ImageComposite_Inp(3MLIB)

Type	Blend Factor [*]	Abbr.
MLIB_BLEND_SRC_ALPHA	(As,As,As,As)	SA
MLIB_BLEND_ONE_MINUS_DST_ALPHA	(1,1,1,1)-(Ad,Ad,Ad,Ad)	OMDA
MLIB_BLEND_ONE_MINUS_SRC_ALPHA	(1,1,1,1)-(As,As,As,As)	OMSA
MLIB_BLEND_SRC_ALPHA_SATURATE	(f,f,f,1)	SAS

[*]: The components of the first source image pixel are (Rd, Gd, Bd, Ad), and the components of the second source pixel are (Rs, Gs, Bs, As). Function $f = \min(As, 1 - Ad)$. The first source image is also the destination image.

The blending formula for in-place processing is:

$$Cd = Cd * D + Cs * S$$

where Cd is the destination pixel (Rd, Gd, Bd, Ad), Cs is the source pixel (Rs, Gs, Bs, As), and D and S are the blend factors for the destination and source, respectively.

PARAMETERS

The function takes the following arguments:

- src1dst* Pointer to the first source and the destination image.
- src2* Pointer to the second source image.
- bsrc1* Blend factor type for the first source image.
- bsrc2* Blend factor type for the second source image.
- cmask* Channel mask to indicate the alpha channel. Each bit of the mask represents a channel in the image. The channel corresponding to the 1 bit is the alpha channel. *cmask* must be either 0x01 or 0x08.

RETURN VALUES

The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

`mllib_ImageBlend_BSRC1_BSRC2(3MLIB)`,
`mllib_ImageBlend_BSRC1_BSRC2_Inp(3MLIB)`, `mllib_ImageComposite(3MLIB)`,
`attributes(5)`

NAME	mllib_ImageConstAdd – addition with a constant						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageConstAdd(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, const mllib_s32 *<i>c</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageConstAdd()</code> function adds a constant to an image on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $dst[x][y][i] = c[i] + src[x][y][i]$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>c</i> An array of constants to be added to each pixel by channel. <code>c[i]</code> contains the constant for channel <code>i</code>.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageConstAdd_Fp(3MLIB)</code> , <code>mllib_ImageConstAdd_Fp_Inp(3MLIB)</code> , <code>mllib_ImageConstAdd_Inp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageConstAdd_Fp(3MLIB)

NAME	mllib_ImageConstAdd_Fp – addition with a constant						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageConstAdd_Fp(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, const mllib_d64 *<i>c</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageConstAdd_Fp()</code> function adds a constant to a floating-point image on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $\text{dst}[x][y][i] = c[i] + \text{src}[x][y][i]$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>c</i> An array of constants to be added to each pixel by channel. <code>c[i]</code> contains the constant for channel <code>i</code>.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageConstAdd(3MLIB)</code> , <code>mllib_ImageConstAdd_Fp_Inp(3MLIB)</code> , <code>mllib_ImageConstAdd_Inp(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageConstAdd_Fp_Inp – addition with a constant

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageConstAdd_Fp_Inp(mllib_image *srcdst, const
    mllib_d64 *c);
```

DESCRIPTION The `mllib_ImageConstAdd_Fp_Inp()` function adds a constant to a floating-point image on a pixel-by-pixel basis, in place.

It uses the following equation:

$$\text{srcdst}[x][y][i] = c[i] + \text{srcdst}[x][y][i]$$

PARAMETERS The function takes the following arguments:

srcdst Pointer to source and destination image.

c An array of constants to be added to each pixel by channel. `c[i]` contains the constant for channel *i*.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageConstAdd(3MLIB)`, `mllib_ImageConstAdd_Fp(3MLIB)`, `mllib_ImageConstAdd_Inp(3MLIB)`, `attributes(5)`

mllib_ImageConstAdd_Inp(3MLIB)

NAME | mllib_ImageConstAdd_Inp – addition with a constant

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageConstAdd_Inp(mllib_image *srcdst, const
mllib_s32 *c);
```

DESCRIPTION | The `mllib_ImageConstAdd_Inp()` function adds a constant to an image on a pixel-by-pixel basis, in place.

It uses the following equation:

$$\text{srcdst}[x][y][i] = c[i] + \text{srcdst}[x][y][i]$$

PARAMETERS | The function takes the following arguments:

srcdst | Pointer to source and destination image.

c | An array of constants to be added to each pixel by channel. `c[i]` contains the constant for channel *i*.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageConstAdd(3MLIB)`, `mllib_ImageConstAdd_Fp(3MLIB)`, `mllib_ImageConstAdd_Fp_Inp(3MLIB)`, `attributes(5)`

NAME	mlib_ImageConstAnd – And with a constant						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmlib [<i>library...</i>] #include <mlib.h> mlib_status mlib_ImageConstAnd(mlib_image *<i>dst</i>, const mlib_image *<i>src</i>, const mlib_s32 *<i>c</i>);</pre>						
DESCRIPTION	<p>The <code>mlib_ImageConstAnd()</code> function computes the And of the source image with a constant.</p> <p>It uses the following equation:</p> $dst[x][y][i] = c[i] \& src[x][y][i]$ <p>The data type of the images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>c</i> Array of constants to be applied to each pixel. <code>c[i]</code> contains the constant for channel <code>i</code>.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mlib_ImageConstAnd_Inp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageConstAnd_Inp(3MLIB)

NAME	mllib_ImageConstAnd_Inp – And with a constant						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageConstAnd_Inp(mllib_image *srcdst, const mllib_s32 *c);</pre>						
DESCRIPTION	<p>The mllib_ImageConstAnd_Inp() function computes the And of the source image with a constant, in place.</p> <p>It uses the following equation:</p> $\text{srcdst}[x][y][i] = c[i] \ \& \ \text{srcdst}[x][y][i]$ <p>The data type of the images can be MLIB_BIT, MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, or MLIB_INT.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>srcdst</i> Pointer to first source and destination image.</p> <p><i>c</i> Array of constants to be applied to each pixel. <i>c</i>[<i>i</i>] contains the constant for channel <i>i</i>.</p>						
RETURN VALUES	The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_ImageConstAnd(3MLIB) , attributes(5)						

NAME mllib_ImageConstAndNot – AndNot with a constant

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageConstAndNot(mllib_image *dst, const
    mllib_image *src, const mllib_s32 *c);
```

DESCRIPTION The mllib_ImageConstAndNot() function computes the And of the Not of the source image with a constant.

It uses the following equation:

$$dst[x][y][i] = c[i] \ \& \ (\sim src[x][y][i])$$

The data type of the images can be MLIB_BIT, MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, or MLIB_INT.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

c Array of constants to be applied to each pixel. c[i] contains the constant for channel i.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageConstAndNot_Inp(3MLIB), attributes(5)

mllib_ImageConstAndNot_Inp(3MLIB)

NAME	mllib_ImageConstAndNot_Inp – AndNot with a constant						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConstAndNot_Inp(mllib_image *srcdst, const mllib_s32 *c);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageConstAndNot_Inp()</code> function computes the And of the Not of the source image with a constant, in place.</p> <p>It uses the following equation:</p> $\text{srcdst}[x][y][i] = c[i] \ \& \ (\sim\text{srcdst}[x][y][i])$ <p>The data type of the images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>srcdst</i> Pointer to first source and destination image.</p> <p><i>c</i> Array of constants to be applied to each pixel. <code>c[i]</code> contains the constant for channel <i>i</i>.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageConstAndNot(3MLIB)</code> , <code>attributes(5)</code>						

NAME | mllib_ImageConstDiv – division into a constant

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageConstDiv(mllib_image *dst, const mllib_image
    *src, const mllib_d64 *c);
```

DESCRIPTION | The mllib_ImageConstDiv() function divides each pixel in an image into a constant value on a pixel-by-pixel basis.

It uses the following equation:

$$dst[x][y][i] = c[i] / src[x][y][i]$$

In the case of $src[x][y][i] = 0$,

$$dst[x][y][i] = 0 \quad \text{if } c[i] = 0$$

$$dst[x][y][i] = DATA_TYPE_MAX \quad \text{if } c[i] > 0$$

$$dst[x][y][i] = DATA_TYPE_MIN \quad \text{if } c[i] < 0$$

where DATA_TYPE is MLIB_U8, MLIB_S16, MLIB_U16, or MLIB_S32 for an image of type MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, or MLIB_INT, respectively.

PARAMETERS | The function takes the following arguments:

dst | Pointer to destination image.

src | Pointer to source image.

c | Constant into which each pixel is divided. *c*[*i*] contains the constant for channel *i*.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_ImageConstDiv_Fp(3MLIB), mllib_ImageConstDiv_Fp_Inp(3MLIB), mllib_ImageConstDiv_Inp(3MLIB), mllib_ImageConstDivShift(3MLIB), mllib_ImageConstDivShift_Inp(3MLIB), attributes(5)

mllib_ImageConstDiv_Fp(3MLIB)

NAME | mllib_ImageConstDiv_Fp – division into a constant

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageConstDiv_Fp(mllib_image *dst, const
    mllib_image *src, const mllib_d64 *c);
```

DESCRIPTION | The mllib_ImageConstDiv_Fp() function divides each pixel in a floating-point image by a constant value on a pixel-by-pixel basis.

It uses the following equation:

$$\text{dst}[x][y][i] = \text{c}[i] / \text{src}[x][y][i]$$

where the operation follows the IEEE-754 standard.

PARAMETERS | The function takes the following arguments:

dst | Pointer to destination image.

src | Pointer to source image.

c | Constant into which each pixel is divided. *c*[*i*] contains the constant for channel *i*.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_ImageConstDiv(3MLIB), mllib_ImageConstDiv_Fp_Inp(3MLIB), mllib_ImageConstDiv_Inp(3MLIB), mllib_ImageConstDivShift(3MLIB), mllib_ImageConstDivShift_Inp(3MLIB), attributes(5)

NAME | mllib_ImageConstDiv_Fp_Inp – division into a constant

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageConstDiv_Fp_Inp(mllib_image *srcdst, const
mllib_d64 *c);
```

DESCRIPTION | The mllib_ImageConstDiv_Fp_Inp() function divides each pixel in a floating-point image by a constant value on a pixel-by-pixel basis, in place. It uses the following equation:

$$\text{srcdst}[x][y][i] = c[i] / \text{srcdst}[x][y][i]$$

where the operation follows the IEEE-754 standard.

PARAMETERS | The function takes the following arguments:

srcdst | Pointer to source and destination image.

c | Constant into which each pixel is divided. *c*[*i*] contains the constant for channel *i*.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_ImageConstDiv(3MLIB), mllib_ImageConstDiv_Fp(3MLIB), mllib_ImageConstDiv_Inp(3MLIB), mllib_ImageConstDivShift(3MLIB), mllib_ImageConstDivShift_Inp(3MLIB), attributes(5)

mllib_ImageConstDiv_Inp(3MLIB)

NAME	mllib_ImageConstDiv_Inp – division into a constant						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConstDiv_Inp(mllib_image *srcdst, const mllib_d64 *c);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageConstDiv_Inp()</code> function divides each pixel in an image by a constant value on a pixel-by-pixel basis, in place.</p> <p>It uses the following equation:</p> $\text{srcdst}[x][y][i] = c[i] / \text{srcdst}[x][y][i]$ <p>In the case of $\text{srcdst}[x][y][i] = 0$,</p> $\begin{aligned} \text{srcdst}[x][y][i] &= 0 && \text{if } c[i] = 0 \\ \text{srcdst}[x][y][i] &= \text{DATA_TYPE_MAX} && \text{if } c[i] > 0 \\ \text{srcdst}[x][y][i] &= \text{DATA_TYPE_MIN} && \text{if } c[i] < 0 \end{aligned}$ <p>where <code>DATA_TYPE</code> is <code>MLIB_U8</code>, <code>MLIB_S16</code>, <code>MLIB_U16</code>, or <code>MLIB_S32</code> for an image of type <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>, respectively.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>srcdst</i></td><td>Pointer to source and destination image.</td></tr><tr><td><i>c</i></td><td>Constant into which each pixel is divided. <code>c[i]</code> contains the constant for channel <i>i</i>.</td></tr></table>	<i>srcdst</i>	Pointer to source and destination image.	<i>c</i>	Constant into which each pixel is divided. <code>c[i]</code> contains the constant for channel <i>i</i> .		
<i>srcdst</i>	Pointer to source and destination image.						
<i>c</i>	Constant into which each pixel is divided. <code>c[i]</code> contains the constant for channel <i>i</i> .						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageConstDiv(3MLIB)</code> , <code>mllib_ImageConstDiv_Fp(3MLIB)</code> , <code>mllib_ImageConstDiv_Fp_Inp(3MLIB)</code> , <code>mllib_ImageConstDivShift(3MLIB)</code> , <code>mllib_ImageConstDivShift_Inp(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageConstDivShift – division into a constant, with shifting

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageConstDivShift(mllib_image *dst, const
    mllib_image *src, const mllib_s32 *c, mllib_s32 shift);
```

DESCRIPTION The `mllib_ImageConstDivShift()` function divides each pixel in an image into a constant value on a pixel-by-pixel basis. It scales the result by a left shift and writes the result to the destination image on a pixel-by-pixel basis.

It uses the following equation:

$$dst[x][y][i] = c[i] / src[x][y][i] * 2^{**shift}$$

In the case of $src[x][y][i] = 0$,

$$dst[x][y][i] = 0 \quad \text{if } c[i] = 0$$

$$dst[x][y][i] = DATA_TYPE_MAX \quad \text{if } c[i] > 0$$

$$dst[x][y][i] = DATA_TYPE_MIN \quad \text{if } c[i] < 0$$

where `DATA_TYPE` is `MLIB_U8`, `MLIB_S16`, `MLIB_U16`, or `MLIB_S32` for an image of type `MLIB_BYTE`, `MLIB_SHORT`, `MLIB_USHORT`, or `MLIB_INT`, respectively.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

c Constant into which each pixel is divided. `c[i]` contains the constant for channel *i*.

shift Left shifting factor. $0 \leq shift \leq 31$.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageConstDiv(3MLIB)`, `mllib_ImageConstDiv_Fp(3MLIB)`, `mllib_ImageConstDiv_Fp_Inp(3MLIB)`, `mllib_ImageConstDiv_Inp(3MLIB)`, `mllib_ImageConstDivShift_Inp(3MLIB)`, `attributes(5)`

mllib_ImageConstDivShift_Inp(3MLIB)

NAME	mllib_ImageConstDivShift_Inp – division into a constant, with shifting						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConstDivShift_Inp(mllib_image *srcdst, const mllib_s32 *c, mllib_s32 shift);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageConstDivShift_Inp()</code> function divides each pixel in an image into a constant value on a pixel-by-pixel basis, in place. It scales the result by a left shift and writes the result to the image on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $\text{srcdst}[x][y][i] = c[i] / \text{srcdst}[x][y][i] * 2^{**\text{shift}}$ <p>In the case of <code>srcdst[x][y][i] = 0</code>,</p> $\begin{aligned} \text{srcdst}[x][y][i] &= 0 && \text{if } c[i] = 0 \\ \text{srcdst}[x][y][i] &= \text{DATA_TYPE_MAX} && \text{if } c[i] > 0 \\ \text{srcdst}[x][y][i] &= \text{DATA_TYPE_MIN} && \text{if } c[i] < 0 \end{aligned}$ <p>where <code>DATA_TYPE</code> is <code>MLIB_U8</code>, <code>MLIB_S16</code>, <code>MLIB_U16</code>, or <code>MLIB_S32</code> for an image of type <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>, respectively.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>srcdst</i> Pointer to source and destination image.</p> <p><i>c</i> Constant into which each pixel is divided. <code>c[i]</code> contains the constant for channel <code>i</code>.</p> <p><i>shift</i> Left shifting factor. $0 \leq \text{shift} \leq 31$.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageConstDiv(3MLIB)</code> , <code>mllib_ImageConstDiv_Fp(3MLIB)</code> , <code>mllib_ImageConstDiv_Fp_Inp(3MLIB)</code> , <code>mllib_ImageConstDiv_Inp(3MLIB)</code> , <code>mllib_ImageConstDivShift(3MLIB)</code> , <code>attributes(5)</code>						

NAME	mlib_ImageConstMul – multiplication with a constant						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmlib [<i>library...</i>] #include <mlib.h> mlib_status mlib_ImageConstMul(mlib_image *<i>dst</i>, const mlib_image *<i>src</i>, const mlib_d64 *<i>c</i>);</pre>						
DESCRIPTION	<p>The <code>mlib_ImageConstMul()</code> function multiplies each pixel in an image by a constant value on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $dst[x][y][i] = c[i] * src[x][y][i]$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>c</i> Constant by which each pixel is multiplied. <code>c[i]</code> contains the constant for channel <code>i</code>.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mlib_ImageConstMul_Fp(3MLIB)</code> , <code>mlib_ImageConstMul_Fp_Inp(3MLIB)</code> , <code>mlib_ImageConstMul_Inp(3MLIB)</code> , <code>mlib_ImageConstMulShift(3MLIB)</code> , <code>mlib_ImageConstMulShift_Inp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageConstMul_Fp(3MLIB)

NAME	mllib_ImageConstMul_Fp – multiply with a constant						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConstMul_Fp(mllib_image *dst, const mllib_image *src, const mllib_d64 *c);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageConstMul_Fp()</code> function multiplies each pixel in a floating-point image by a constant value on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $\text{dst}[x][y][i] = c[i] * \text{src}[x][y][i]$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>c</i> Constant by which each pixel is multiplied. <code>c[i]</code> contains the constant for channel <code>i</code>.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageConstMul(3MLIB)</code> , <code>mllib_ImageConstMul_Fp_Inp(3MLIB)</code> , <code>mllib_ImageConstMul_Inp(3MLIB)</code> , <code>mllib_ImageConstMulShift(3MLIB)</code> , <code>mllib_ImageConstMulShift_Inp(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageConstMul_Fp_Inp – multiply with a constant

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageConstMul_Fp_Inp(mllib_image *srcdst, const
mllib_d64 *c);
```

DESCRIPTION The mllib_ImageConstMul_Fp_Inp() function multiplies each pixel in a floating-point image by a constant value on a pixel-by-pixel basis, in place. It uses the following equation:

$$\text{srcdst}[x][y][i] = c[i] * \text{srcdst}[x][y][i]$$

PARAMETERS The function takes the following arguments:

srcdst Pointer to source and destination image.

c Constant by which each pixel is multiplied. *c*[*i*] contains the constant for channel *i*.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageConstMul(3MLIB), mllib_ImageConstMul_Fp(3MLIB), mllib_ImageConstMul_Inp(3MLIB), mllib_ImageConstMulShift(3MLIB), mllib_ImageConstMulShift_Inp(3MLIB), attributes(5)

mllib_ImageConstMul_Inp(3MLIB)

NAME	mllib_ImageConstMul_Inp – multiply with a constant						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConstMul_Inp(mllib_image *srcdst, const mllib_d64 *c);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageConstMul_Inp()</code> function multiplies each pixel in an image by a constant value on a pixel-by-pixel basis, in place.</p> <p>It uses the following equation:</p> $\text{srcdst}[x][y][i] = c[i] * \text{srcdst}[x][y][i]$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>srcdst</i> Pointer to source and destination image.</p> <p><i>c</i> Constant by which each pixel is multiplied. <code>c[i]</code> contains the constant for channel <i>i</i>.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageConstMul(3MLIB)</code> , <code>mllib_ImageConstMul_Fp(3MLIB)</code> , <code>mllib_ImageConstMul_Fp_Inp(3MLIB)</code> , <code>mllib_ImageConstMulShift(3MLIB)</code> , <code>mllib_ImageConstMulShift_Inp(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageConstMulShift – multiply with a constant, with shifting

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageConstMulShift(mllib_image *dst, const
    mllib_image *src, const mllib_s32 *c, mllib_s32 shift);
```

DESCRIPTION The `mllib_ImageConstMulShift()` function multiplies each pixel in an image by a constant value on a pixel-by-pixel basis. It scales the result by a right shift and writes the result to the destination image on a pixel-by-pixel basis.

It uses the following equation:

$$dst[x][y][i] = c[i] * src[x][y][i] * 2^{*(-shift)}$$

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

c Constant by which each pixel is multiplied. `c[i]` contains the constant for channel *i*.

shift Right shifting factor. $0 \leq shift \leq 31$.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageConstMul(3MLIB)`, `mllib_ImageConstMul_Fp(3MLIB)`, `mllib_ImageConstMul_Fp_Inp(3MLIB)`, `mllib_ImageConstMul_Inp(3MLIB)`, `mllib_ImageConstMulShift_Inp(3MLIB)`, `attributes(5)`

mllib_ImageConstMulShift_Inp(3MLIB)

NAME	mllib_ImageConstMulShift_Inp – multiply with a constant, with shifting						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConstMulShift_Inp(mllib_image *srcdst, const mllib_s32 *c, mllib_s32 shift);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageConstMulShift_Inp()</code> function multiplies each pixel in an image by a constant value on a pixel-by-pixel basis. It scales the result by a right shift and writes the result to the destination image on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $\text{srcdst}[x][y][i] = c[i] * \text{srcdst}[x][y][i] * 2^{**(-\text{shift})}$						
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>srcdst</i></td><td>Pointer to source and destination image.</td></tr><tr><td><i>c</i></td><td>Constant by which each pixel is multiplied. <code>c[i]</code> contains the constant for channel <code>i</code>.</td></tr><tr><td><i>shift</i></td><td>Right shifting factor. $0 \leq \text{shift} \leq 31$.</td></tr></table>	<i>srcdst</i>	Pointer to source and destination image.	<i>c</i>	Constant by which each pixel is multiplied. <code>c[i]</code> contains the constant for channel <code>i</code> .	<i>shift</i>	Right shifting factor. $0 \leq \text{shift} \leq 31$.
<i>srcdst</i>	Pointer to source and destination image.						
<i>c</i>	Constant by which each pixel is multiplied. <code>c[i]</code> contains the constant for channel <code>i</code> .						
<i>shift</i>	Right shifting factor. $0 \leq \text{shift} \leq 31$.						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageConstMul(3MLIB)</code> , <code>mllib_ImageConstMul_Fp(3MLIB)</code> , <code>mllib_ImageConstMul_Fp_Inp(3MLIB)</code> , <code>mllib_ImageConstMul_Inp(3MLIB)</code> , <code>mllib_ImageConstMulShift(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageConstNotAnd – NotAnd with a constant

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageConstNotAnd(mllib_image *dst, const
    mllib_image *src, const mllib_s32 *c);
```

DESCRIPTION The `mllib_ImageConstNotAnd()` function computes the logical And of the source image with a constant and then takes the logical Not of that result on a pixel-by-pixel basis.

It uses the following equation:

$$dst[x][y][i] = \sim(c[i] \& src[x][y][i])$$

The data type of the images can be `MLIB_BIT`, `MLIB_BYTE`, `MLIB_SHORT`, `MLIB_USHORT`, or `MLIB_INT`.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

c Array of constants to be applied to each pixel. `c[i]` contains the constant for channel *i*.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageConstNotAnd_Inp(3MLIB)`, `attributes(5)`

mllib_ImageConstNotAnd_Inp(3MLIB)

NAME	mllib_ImageConstNotAnd_Inp – NotAnd with a constant, in place						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConstNotAnd_Inp(mllib_image *srcdst, const mllib_s32 *c);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageConstNotAnd_Inp()</code> function computes the logical And of the source image with a constant and then takes the logical Not of that result on a pixel-by-pixel basis, in place.</p> <p>It uses the following equation:</p> $\text{srcdst}[x][y][i] = \sim(c[i] \& \text{srcdst}[x][y][i])$ <p>The data type of the images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>srcdst</i> Pointer to first source and destination image.</p> <p><i>c</i> Array of constants to be applied to each pixel. <i>c</i>[<i>i</i>] contains the constant for channel <i>i</i>.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageConstNotAnd(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageConstNotOr – NotOr with a constant

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageConstNotOr(mllib_image *dst, const
    mllib_image *src, const mllib_s32 *c);
```

DESCRIPTION The `mllib_ImageConstNotOr()` function computes the logical Or of the source image with a constant and then takes the logical Not of that result on a pixel-by-pixel basis.

It uses the following equation:

$$dst[x][y][i] = \sim(c[i] \mid src[x][y][i])$$

The data type of the images can be `MLIB_BIT`, `MLIB_BYTE`, `MLIB_SHORT`, `MLIB_USHORT`, or `MLIB_INT`.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

c Array of constants to be applied to each pixel. `c[i]` contains the constant for channel *i*.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageConstNotOr_Inp(3MLIB)`, `attributes(5)`

mllib_ImageConstNotOr_Inp(3MLIB)

NAME	mllib_ImageConstNotOr_Inp – NotOr with a constant, in place						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageConstNotOr_Inp(mllib_image *<i>srcdst</i>, const mllib_s32 *<i>c</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageConstNotOr_Inp()</code> function computes the logical Or of the source image with a constant and then takes the logical Not of that result on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $\text{srcdst}[x][y][i] = \sim(c[i] \mid \text{srcdst}[x][y][i])$ <p>The data type of the images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>srcdst</i> Pointer to first source and destination image.</p> <p><i>c</i> Array of constants to be applied to each pixel. <code>c[i]</code> contains the constant for channel <i>i</i>.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageConstNotOr(3MLIB)</code> , <code>attributes(5)</code>						

NAME	mllib_ImageConstNotXor – NotXor with a constant						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConstNotXor(mllib_image *dst, const mllib_image *src, const mllib_s32 *c);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageConstNotXor()</code> function computes the logical exclusive Or of the source image with a constant and then takes the logical Not of that result on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sim(c[i] \wedge src[x][y][i])$ <p>The data type of the images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>c</i> Array of constants to be applied to each pixel. <code>c[i]</code> contains the constant for channel <i>i</i>.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageConstNotXor_Inp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageConstNotXor_Inp(3MLIB)

NAME	mllib_ImageConstNotXor_Inp – NotXor with a constant, in place						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConstNotXor_Inp(mllib_image *srcdst, const mllib_s32 *c);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageConstNotXor_Inp()</code> function computes the logical exclusive Or of the source image with a constant and then takes the logical Not of that result on a pixel-by-pixel basis, in place.</p> <p>It uses the following equation:</p> $\text{srcdst}[x][y][i] = \sim(c[i] \wedge \text{srcdst}[x][y][i])$ <p>The data type of the images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>srcdst</i> Pointer to first source and destination image.</p> <p><i>c</i> Array of constants to be applied to each pixel. <code>c[i]</code> contains the constant for channel <i>i</i>.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageConstNotXor(3MLIB)</code> , <code>attributes(5)</code>						

NAME	mllib_ImageConstOr – Or with a constant						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConstOr(mllib_image *dst, const mllib_image *src, const mllib_s32 *c);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageConstOr()</code> function computes the logical Or of the source image with a constant on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $dst[x][y][i] = c[i] src[x][y][i]$ <p>The data type of the images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>c</i> Array of constants to be applied to each pixel. <code>c[i]</code> contains the constant for channel <i>i</i>.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageConstOr_Inp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageConstOr_Inp(3MLIB)

NAME | mllib_ImageConstOr_Inp – Or with a constant, in place

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageConstOr_Inp(mllib_image *srcdst, const
    mllib_s32 *c);
```

DESCRIPTION | The `mllib_ImageConstOr_Inp()` function computes the logical Or of the source image with a constant on a pixel-by-pixel basis, in place.

It uses the following equation:

$$\text{srcdst}[x][y][i] = c[i] \mid \text{srcdst}[x][y][i]$$

The data type of the images can be `MLIB_BIT`, `MLIB_BYTE`, `MLIB_SHORT`, `MLIB_USHORT`, or `MLIB_INT`.

PARAMETERS | The function takes the following arguments:

srcdst | Pointer to first source and destination image.

c | Array of constants to be applied to each pixel. `c[i]` contains the constant for channel *i*.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_ImageConstOr\(3MLIB\)](#), [attributes\(5\)](#)

NAME	mllib_ImageConstOrNot – OrNot with a constant						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConstOrNot(mllib_image *dst, const mllib_image *src, const mllib_s32 *c);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageConstOrNot()</code> function computes the logical Not of the source image and then takes the logical Or of that result with a constant on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $dst[x][y][i] = c[i] (\sim src[x][y][i])$ <p>The data type of the images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>c</i> Array of constants to be applied to each pixel. <code>c[i]</code> contains the constant for channel <i>i</i>.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageConstOrNot_Inp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageConstOrNot_Inp(3MLIB)

NAME	mllib_ImageConstOrNot_Inp – OrNot with a constant, in place						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageConstOrNot_Inp(mllib_image *<i>srcdst</i>, const mllib_s32 *<i>c</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageConstOrNot_Inp()</code> function computes the logical Not of the source image and then takes the logical Or of that result with a constant on a pixel-by-pixel basis, in place.</p> <p>It uses the following equation:</p> $\text{srcdst}[x][y][i] = c[i] \mid (\sim \text{srcdst}[x][y][i])$ <p>The data type of the images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>srcdst</i></td><td>Pointer to first source and destination image.</td></tr><tr><td><i>c</i></td><td>Array of constants to be applied to each pixel. <code>c[i]</code> contains the constant for channel <code>i</code>.</td></tr></table>	<i>srcdst</i>	Pointer to first source and destination image.	<i>c</i>	Array of constants to be applied to each pixel. <code>c[i]</code> contains the constant for channel <code>i</code> .		
<i>srcdst</i>	Pointer to first source and destination image.						
<i>c</i>	Array of constants to be applied to each pixel. <code>c[i]</code> contains the constant for channel <code>i</code> .						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageConstOrNot(3MLIB)</code> , <code>attributes(5)</code>						

NAME	mlib_ImageConstSub – Subtraction with a constant						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmlib [<i>library...</i>] #include <mlib.h> mlib_status mlib_ImageConstSub(mlib_image *<i>dst</i>, const mlib_image *<i>src</i>, const mlib_s32 *<i>c</i>);</pre>						
DESCRIPTION	<p>The <code>mlib_ImageConstSub()</code> function subtracts an image pixel from a constant on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $dst[x][y][i] = c[i] - src[x][y][i]$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>c</i> Array of constants from which each pixel is subtracted by channel. <code>c[i]</code> contains the constant for channel <i>i</i>.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mlib_ImageConstSub_Fp(3MLIB)</code> , <code>mlib_ImageConstSub_Fp_Inp(3MLIB)</code> , <code>mlib_ImageConstSub_Inp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageConstSub_Fp(3MLIB)

NAME | mllib_ImageConstSub_Fp – Subtraction with a constant

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
mllib_status mllib_ImageConstSub_Fp(mllib_image *dst, const  
mllib_image *src, const mllib_d64 *c);
```

DESCRIPTION | The `mllib_ImageConstSub_Fp()` function subtracts a floating-point image pixel from a constant on a pixel-by-pixel basis.

It uses the following equation:

$$dst[x][y][i] = c[i] - src[x][y][i]$$

PARAMETERS | The function takes the following arguments:

dst | Pointer to destination image.

src | Pointer to source image.

c | Array of constants from which each pixel is subtracted by channel. `c[i]` contains the constant for channel `i`.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageConstSub(3MLIB)`, `mllib_ImageConstSub_Fp_Inp(3MLIB)`, `mllib_ImageConstSub_Inp(3MLIB)`, `attributes(5)`

NAME mllib_ImageConstSub_Fp_Inp – subtraction with a constant

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageConstSub_Fp_Inp(mllib_image *srcdst, const
mllib_d64 *c);
```

DESCRIPTION The mllib_ImageConstSub_Fp_Inp() function subtracts a floating-point image pixel from a constant on a pixel-by-pixel basis, in place.

It uses the following equation:

$$srcdst[x][y][i] = c[i] - srcdst[x][y][i]$$

PARAMETERS The function takes the following arguments:

srcdst Pointer to source and destination image.

c Array of constants from which each pixel is subtracted by channel. *c*[*i*] contains the constant for channel *i*.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageConstSub(3MLIB), mllib_ImageConstSub_Fp(3MLIB), mllib_ImageConstSub_Inp(3MLIB), attributes(5)

mllib_ImageConstSub_Inp(3MLIB)

NAME | mllib_ImageConstSub_Inp – Subtraction with a constant

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageConstSub_Inp(mllib_image *srcdst, const
    mllib_s32 *c);
```

DESCRIPTION | The `mllib_ImageConstSub_Inp()` function subtracts an image pixel from a constant on a pixel-by-pixel basis, in place.

It uses the following equation:

$$\text{srcdst}[x][y][i] = c[i] - \text{srcdst}[x][y][i]$$

PARAMETERS | The function takes the following arguments:

srcdst | Pointer to source and destination image.

c | Array of constants from which each pixel is subtracted by channel. `c[i]` contains the constant for channel *i*.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageConstSub(3MLIB)`, `mllib_ImageConstSub_Fp(3MLIB)`, `mllib_ImageConstSub_Fp_Inp(3MLIB)`, `attributes(5)`

NAME mllib_ImageConstXor – Xor with a constant

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageConstXor(mllib_image *dst, const mllib_image
    *src, const mllib_s32 *c);
```

DESCRIPTION The `mllib_ImageConstXor()` function computes the exclusive Or of the source image with a constant on a pixel-by-pixel basis.

It uses the following equation:

$$dst[x][y][i] = c[i] \wedge src[x][y][i]$$

The data type of the images can be `MLIB_BIT`, `MLIB_BYTE`, `MLIB_SHORT`, `MLIB_USHORT`, or `MLIB_INT`.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

c Array of constants to be applied to each pixel. `c[i]` contains the constant for channel *i*.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageConstXor_Inp(3MLIB)`, `attributes(5)`

mllib_ImageConstXor_Inp(3MLIB)

NAME | mllib_ImageConstXor_Inp – Xor with a constant, in place

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageConstXor_Inp(mllib_image *srcdst, const
mllib_s32 *c);
```

DESCRIPTION | The mllib_ImageConstXor_Inp() function computes the exclusive Or of the source image with a constant on a pixel-by-pixel basis, in place.

It uses the following equation:

$$\text{srcdst}[x][y][i] = c[i] \wedge \text{srcdst}[x][y][i]$$

The data type of the images can be MLLIB_BIT, MLLIB_BYTE, MLLIB_SHORT, MLLIB_USHORT, or MLLIB_INT.

PARAMETERS | The function takes the following arguments:

srcdst | Pointer to first source and destination image.

c | Array of constants to be applied to each pixel. *c*[*i*] contains the constant for channel *i*.

RETURN VALUES | The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_ImageConstXor\(3MLIB\)](#), [attributes\(5\)](#)

NAME	mllib_ImageConv2x2 – 2x2 convolution
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConv2x2(mllib_image *dst, const mllib_image *src, const mllib_s32 *kernel, mllib_s32 scale, mllib_s32 cmask, mllib_edge edge);</pre>
DESCRIPTION	<p>The <code>mllib_ImageConv2x2()</code> function performs a 2x2 convolution on the source image by using the user-supplied kernel.</p> <p>The input image and the output image must have the same image type and have the same number of channels. The unselected channels in the output image are not overwritten. For single-channel images, the channel mask is ignored.</p> <p>For this convolution, the key element of the convolution kernel is located at (0, 0) of the kernel matrix.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sum_{p=-dm}^{m-1-dm} \sum_{q=-dn}^{n-1-dn} src[x+p][y+q][i] * k[p][q] * 2^{**}(-scale)$ <p>where $m = 2$, $n = 2$, $dm = (m - 1) / 2 = 0$, $dn = (n - 1) / 2 = 0$.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>kernel</i> Pointer to the convolution kernel, in row major order.</p> <p><i>scale</i> Scaling factor.</p> <p><i>cmask</i> Channel mask to indicate the channels to be convolved, each bit of which represents a channel in the image. The channels corresponding to bits with a value of 1 are those to be processed. For a single-channel image, the channel mask is ignored.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SRC_EXTEND</pre>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .

mllib_ImageConv2x2(3MLIB)

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageConv2x2_Fp(3MLIB)`, `mllib_ImageConv2x2Index(3MLIB)`,
`mllib_ImageConv3x3(3MLIB)`, `mllib_ImageConv3x3_Fp(3MLIB)`,
`mllib_ImageConv3x3Index(3MLIB)`, `mllib_ImageConv4x4(3MLIB)`,
`mllib_ImageConv4x4_Fp(3MLIB)`, `mllib_ImageConv4x4Index(3MLIB)`,
`mllib_ImageConv5x5(3MLIB)`, `mllib_ImageConv5x5_Fp(3MLIB)`,
`mllib_ImageConv5x5Index(3MLIB)`, `mllib_ImageConv7x7(3MLIB)`,
`mllib_ImageConv7x7_Fp(3MLIB)`, `mllib_ImageConv7x7Index(3MLIB)`,
`mllib_ImageConvKernelConvert(3MLIB)`, `mllib_ImageConvMxN(3MLIB)`,
`mllib_ImageConvMxN_Fp(3MLIB)`, `mllib_ImageConvMxNIndex(3MLIB)`,
`mllib_ImageConvolveMxN(3MLIB)`, `mllib_ImageConvolveMxN_Fp(3MLIB)`,
`mllib_ImageSConv3x3(3MLIB)`, `mllib_ImageSConv3x3_Fp(3MLIB)`,
`mllib_ImageSConv5x5(3MLIB)`, `mllib_ImageSConv5x5_Fp(3MLIB)`,
`mllib_ImageSConv7x7(3MLIB)`, `mllib_ImageSConv7x7_Fp(3MLIB)`,
`mllib_ImageSConvKernelConvert(3MLIB)`, `attributes(5)`

NAME	mllib_ImageConv2x2_Fp – 2x2 convolution
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConv2x2_Fp(mllib_image *dst, const mllib_image *src, const mllib_d64 *kernel, mllib_s32 cmask, mllib_edge edge);</pre>
DESCRIPTION	<p>The <code>mllib_ImageConv2x2_Fp()</code> function performs a 2x2 convolution on the source image by using the user-supplied kernel.</p> <p>The input image and the output image must have the same image type and have the same number of channels. The unselected channels in the output image are not overwritten. For single-channel images, the channel mask is ignored.</p> <p>For this convolution, the key element of the convolution kernel is located at (0, 0) of the kernel matrix.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sum_{p=-dm}^{m-1-dm} \sum_{q=-dn}^{n-1-dn} src[x+p][y+q][i] * k[p][q]$ <p>where $m = 2$, $n = 2$, $dm = (m - 1) / 2 = 0$, $dn = (n - 1) / 2 = 0$.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>kernel</i> Pointer to the convolution kernel, in row major order.</p> <p><i>cmask</i> Channel mask to indicate the channels to be convolved, each bit of which represents a channel in the image. The channels corresponding to bits with a value of 1 are those to be processed. For a single-channel image, the channel mask is ignored.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SRC_EXTEND</pre>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:

mllib_ImageConv2x2_Fp(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

mllib_ImageConv2x2(3MLIB), mllib_ImageConv2x2Index(3MLIB),
mllib_ImageConv3x3(3MLIB), mllib_ImageConv3x3_Fp(3MLIB),
mllib_ImageConv3x3Index(3MLIB), mllib_ImageConv4x4(3MLIB),
mllib_ImageConv4x4_Fp(3MLIB), mllib_ImageConv4x4Index(3MLIB),
mllib_ImageConv5x5(3MLIB), mllib_ImageConv5x5_Fp(3MLIB),
mllib_ImageConv5x5Index(3MLIB), mllib_ImageConv7x7(3MLIB),
mllib_ImageConv7x7_Fp(3MLIB), mllib_ImageConv7x7Index(3MLIB),
mllib_ImageConvKernelConvert(3MLIB), mllib_ImageConvMxN(3MLIB),
mllib_ImageConvMxN_Fp(3MLIB), mllib_ImageConvMxNIndex(3MLIB),
mllib_ImageConvolveMxN(3MLIB), mllib_ImageConvolveMxN_Fp(3MLIB),
mllib_ImageSConv3x3(3MLIB), mllib_ImageSConv3x3_Fp(3MLIB),
mllib_ImageSConv5x5(3MLIB), mllib_ImageSConv5x5_Fp(3MLIB),
mllib_ImageSConv7x7(3MLIB), mllib_ImageSConv7x7_Fp(3MLIB),
mllib_ImageSConvKernelConvert(3MLIB), attributes(5)

NAME	mllib_ImageConv2x2Index – 2x2 convolution on a color indexed image
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConv2x2Index(mllib_image *dst, const mllib_image *src, const mllib_s32 *kernel, mllib_s32 scale, mllib_edge edge, const void *colormap);</pre>
DESCRIPTION	<p>The <code>mllib_ImageConv2x2Index()</code> function performs a 2x2 convolution on the color indexed source image by using the user-supplied kernel.</p> <p>The input and output images must have the same image type and size.</p> <p>For this convolution, the key element of the convolution kernel is located at (0, 0) of the kernel matrix.</p> <p>This function performs the convolution on color indexed image. The input image and the output image must be single-channel images. The image type must be <code>MLIB_BYTE</code> or <code>MLIB_SHORT</code>.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sum_{p=-dm}^{m-1-dm} \sum_{q=-dn}^{n-1-dn} src[x+p][y+q][i] * k[p][q] * 2^{**}(-scale)$ <p>where $m = 2$, $n = 2$, $dm = (m - 1) / 2 = 0$, $dn = (n - 1) / 2 = 0$.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>kernel</i> Pointer to the convolution kernel, in row major order.</p> <p><i>scale</i> Scaling factor.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SRC_EXTEND</pre> <p><i>colormap</i> Internal data structure for inverse color mapping. This data structure is generated by the <code>mllib_ImageColorTrue2IndexInit()</code> function.</p>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .

mllib_ImageConv2x2Index(3MLIB)

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageConv2x2(3MLIB)`, `mllib_ImageConv2x2_Fp(3MLIB)`,
`mllib_ImageConv3x3(3MLIB)`, `mllib_ImageConv3x3_Fp(3MLIB)`,
`mllib_ImageConv3x3Index(3MLIB)`, `mllib_ImageConv4x4(3MLIB)`,
`mllib_ImageConv4x4_Fp(3MLIB)`, `mllib_ImageConv4x4Index(3MLIB)`,
`mllib_ImageConv5x5(3MLIB)`, `mllib_ImageConv5x5_Fp(3MLIB)`,
`mllib_ImageConv5x5Index(3MLIB)`, `mllib_ImageConv7x7(3MLIB)`,
`mllib_ImageConv7x7_Fp(3MLIB)`, `mllib_ImageConv7x7Index(3MLIB)`,
`mllib_ImageConvKernelConvert(3MLIB)`, `mllib_ImageConvMxN(3MLIB)`,
`mllib_ImageConvMxN_Fp(3MLIB)`, `mllib_ImageConvMxNIndex(3MLIB)`,
`mllib_ImageConvolveMxN(3MLIB)`, `mllib_ImageConvolveMxN_Fp(3MLIB)`,
`mllib_ImageSConv3x3(3MLIB)`, `mllib_ImageSConv3x3_Fp(3MLIB)`,
`mllib_ImageSConv5x5(3MLIB)`, `mllib_ImageSConv5x5_Fp(3MLIB)`,
`mllib_ImageSConv7x7(3MLIB)`, `mllib_ImageSConv7x7_Fp(3MLIB)`,
`mllib_ImageSConvKernelConvert(3MLIB)`, `attributes(5)`

NAME	mllib_ImageConv3x3 – 3x3 convolution
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConv3x3(mllib_image *dst, const mllib_image *src, const mllib_s32 *kernel, mllib_s32 scale, mllib_s32 cmask, mllib_edge edge);</pre>
DESCRIPTION	<p>The <code>mllib_ImageConv3x3()</code> function performs a 3x3 convolution on the source image by using the user-supplied kernel.</p> <p>The input image and the output image must have the same image type and have the same number of channels. The unselected channels in the output image are not overwritten. For single-channel images, the channel mask is ignored.</p> <p>For this convolution, the key element of the convolution kernel is located at the center of the kernel matrix.</p> <p>The data type of source and destination images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sum_{p=-dm}^{m-1-dm} \sum_{q=-dn}^{n-1-dn} src[x+p][y+q][i] * k[p][q] * 2^{**(-scale)}$ <p>where $m = 3$, $n = 3$, $dm = (m - 1) / 2 = 1$, $dn = (n - 1) / 2 = 1$.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>kernel</i> Pointer to the convolution kernel, in row major order.</p> <p><i>scale</i> Scaling factor.</p> <p><i>cmask</i> Channel mask to indicate the channels to be convolved, each bit of which represents a channel in the image. The channels corresponding to bits with a value of 1 are those to be processed. For a single-channel image, the channel mask is ignored.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SRC_EXTEND</pre>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .

mllib_ImageConv3x3(3MLIB)

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageConv2x2(3MLIB)`, `mllib_ImageConv2x2_Fp(3MLIB)`,
`mllib_ImageConv2x2Index(3MLIB)`, `mllib_ImageConv3x3_Fp(3MLIB)`,
`mllib_ImageConv3x3Index(3MLIB)`, `mllib_ImageConv4x4(3MLIB)`,
`mllib_ImageConv4x4_Fp(3MLIB)`, `mllib_ImageConv4x4Index(3MLIB)`,
`mllib_ImageConv5x5(3MLIB)`, `mllib_ImageConv5x5_Fp(3MLIB)`,
`mllib_ImageConv5x5Index(3MLIB)`, `mllib_ImageConv7x7(3MLIB)`,
`mllib_ImageConv7x7_Fp(3MLIB)`, `mllib_ImageConv7x7Index(3MLIB)`,
`mllib_ImageConvKernelConvert(3MLIB)`, `mllib_ImageConvMxN(3MLIB)`,
`mllib_ImageConvMxN_Fp(3MLIB)`, `mllib_ImageConvMxNIndex(3MLIB)`,
`mllib_ImageConvolveMxN(3MLIB)`, `mllib_ImageConvolveMxN_Fp(3MLIB)`,
`mllib_ImageSConv3x3(3MLIB)`, `mllib_ImageSConv3x3_Fp(3MLIB)`,
`mllib_ImageSConv5x5(3MLIB)`, `mllib_ImageSConv5x5_Fp(3MLIB)`,
`mllib_ImageSConv7x7(3MLIB)`, `mllib_ImageSConv7x7_Fp(3MLIB)`,
`mllib_ImageSConvKernelConvert(3MLIB)`, `attributes(5)`

NAME	mllib_ImageConv3x3_Fp – 3x3 convolution
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConv3x3_Fp(mllib_image *dst, const mllib_image *src, const mllib_d64 *kernel, mllib_s32 cmask, mllib_edge edge);</pre>
DESCRIPTION	<p>The <code>mllib_ImageConv3x3_Fp()</code> function performs a floating-point 3x3 convolution on the source image by using the user-supplied kernel.</p> <p>The input image and the output image must have the same image type and have the same number of channels. The unselected channels in the output image are not overwritten. For single-channel images, the channel mask is ignored.</p> <p>For this convolution, the key element of the convolution kernel is located at the center of the kernel matrix.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sum_{p=-dm}^{m-1-dm} \sum_{q=-dn}^{n-1-dn} src[x+p][y+q][i] * k[p][q]$ <p>where $m = 3$, $n = 3$, $dm = (m - 1) / 2 = 1$, $dn = (n - 1) / 2 = 1$.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>kernel</i> Pointer to the convolution kernel, in row major order.</p> <p><i>cmask</i> Channel mask to indicate the channels to be convolved, each bit of which represents a channel in the image. The channels corresponding to bits with a value of 1 are those to be processed. For a single-channel image, the channel mask is ignored.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SRC_EXTEND</pre>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:

mllib_ImageConv3x3_Fp(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

mllib_ImageConv2x2(3MLIB), mllib_ImageConv2x2_Fp(3MLIB),
mllib_ImageConv2x2Index(3MLIB), mllib_ImageConv3x3(3MLIB),
mllib_ImageConv3x3Index(3MLIB), mllib_ImageConv4x4(3MLIB),
mllib_ImageConv4x4_Fp(3MLIB), mllib_ImageConv4x4Index(3MLIB),
mllib_ImageConv5x5(3MLIB), mllib_ImageConv5x5_Fp(3MLIB),
mllib_ImageConv5x5Index(3MLIB), mllib_ImageConv7x7(3MLIB),
mllib_ImageConv7x7_Fp(3MLIB), mllib_ImageConv7x7Index(3MLIB),
mllib_ImageConvKernelConvert(3MLIB), mllib_ImageConvMxN(3MLIB),
mllib_ImageConvMxN_Fp(3MLIB), mllib_ImageConvMxNIndex(3MLIB),
mllib_ImageConvolveMxN(3MLIB), mllib_ImageConvolveMxN_Fp(3MLIB),
mllib_ImageSConv3x3(3MLIB), mllib_ImageSConv3x3_Fp(3MLIB),
mllib_ImageSConv5x5(3MLIB), mllib_ImageSConv5x5_Fp(3MLIB),
mllib_ImageSConv7x7(3MLIB), mllib_ImageSConv7x7_Fp(3MLIB),
mllib_ImageSConvKernelConvert(3MLIB), attributes(5)

NAME	mllib_ImageConv3x3Index – 3x3 convolution on a color indexed image
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConv3x3Index(mllib_image *dst, const mllib_image *src, const mllib_s32 *kernel, mllib_s32 scale, mllib_edge edge, const void *colormap);</pre>
DESCRIPTION	<p>The <code>mllib_ImageConv3x3Index()</code> function performs a 3x3 convolution on the color indexed source image by using the user-supplied kernel.</p> <p>The input and output images must have the same image type and size.</p> <p>For this convolution, the key element of the convolution kernel is located at the center of the kernel matrix.</p> <p>This function performs the convolution on color indexed image. The input image and the output image must be single-channel images. The image type must be <code>MLIB_BYTE</code> or <code>MLIB_SHORT</code>.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sum_{p=-dm}^{m-1-dm} \sum_{q=-dn}^{n-1-dn} src[x+p][y+q][i] * k[p][q] * 2^{**}(-scale)$ <p>where $m = 3$, $n = 3$, $dm = (m - 1) / 2 = 1$, $dn = (n - 1) / 2 = 1$.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>kernel</i> Pointer to the convolution kernel, in row major order.</p> <p><i>scale</i> Scaling factor.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SRC_EXTEND</pre> <p><i>colormap</i> Internal data structure for inverse color mapping. This data structure is generated by the <code>mllib_ImageColorTrue2IndexInit()</code> function. The source and destination images must be single-channel images.</p>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .

mllib_ImageConv3x3Index(3MLIB)

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageConv2x2(3MLIB), mllib_ImageConv2x2_Fp(3MLIB), mllib_ImageConv2x2Index(3MLIB), mllib_ImageConv3x3(3MLIB), mllib_ImageConv3x3_Fp(3MLIB), mllib_ImageConv4x4(3MLIB), mllib_ImageConv4x4_Fp(3MLIB), mllib_ImageConv4x4Index(3MLIB), mllib_ImageConv5x5(3MLIB), mllib_ImageConv5x5_Fp(3MLIB), mllib_ImageConv5x5Index(3MLIB), mllib_ImageConv7x7(3MLIB), mllib_ImageConv7x7_Fp(3MLIB), mllib_ImageConv7x7Index(3MLIB), mllib_ImageConvKernelConvert(3MLIB), mllib_ImageConvMxN(3MLIB), mllib_ImageConvMxN_Fp(3MLIB), mllib_ImageConvMxNIndex(3MLIB), mllib_ImageConvolveMxN(3MLIB), mllib_ImageConvolveMxN_Fp(3MLIB), mllib_ImageSConv3x3(3MLIB), mllib_ImageSConv3x3_Fp(3MLIB), mllib_ImageSConv5x5(3MLIB), mllib_ImageSConv5x5_Fp(3MLIB), mllib_ImageSConv7x7(3MLIB), mllib_ImageSConv7x7_Fp(3MLIB), mllib_ImageSConvKernelConvert(3MLIB), attributes(5)

NAME	mllib_ImageConv4x4 – 4x4 convolution
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConv4x4(mllib_image *dst, const mllib_image *src, const mllib_s32 *kernel, mllib_s32 scale, mllib_s32 cmask, mllib_edge edge);</pre>
DESCRIPTION	<p>The <code>mllib_ImageConv4x4()</code> function performs a 4x4 convolution on the source image by using the user-supplied kernel.</p> <p>The input image and the output image must have the same image type and have the same number of channels. The unselected channels in the output image are not overwritten. For single-channel images, the channel mask is ignored.</p> <p>For this convolution, the key element of the convolution kernel is located at (1, 1) of the kernel matrix.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sum_{p=-dm}^{m-1-dm} \sum_{q=-dn}^{n-1-dn} src[x+p][y+q][i] * k[p][q] * 2^{**}(-scale)$ <p>where $m = 4$, $n = 4$, $dm = (m - 1) / 2 = 1$, $dn = (n - 1) / 2 = 1$.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>kernel</i> Pointer to the convolution kernel, in row major order.</p> <p><i>scale</i> Scaling factor.</p> <p><i>cmask</i> Channel mask to indicate the channels to be convolved, each bit of which represents a channel in the image. The channels corresponding to bits with a value of 1 are those to be processed. For a single-channel image, the channel mask is ignored.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SRC_EXTEND</pre>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .

mllib_ImageConv4x4(3MLIB)

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageConv2x2(3MLIB), mllib_ImageConv2x2_Fp(3MLIB),
mllib_ImageConv2x2Index(3MLIB), mllib_ImageConv3x3(3MLIB),
mllib_ImageConv3x3_Fp(3MLIB), mllib_ImageConv3x3Index(3MLIB),
mllib_ImageConv4x4_Fp(3MLIB), mllib_ImageConv4x4Index(3MLIB),
mllib_ImageConv5x5(3MLIB), mllib_ImageConv5x5_Fp(3MLIB),
mllib_ImageConv5x5Index(3MLIB), mllib_ImageConv7x7(3MLIB),
mllib_ImageConv7x7_Fp(3MLIB), mllib_ImageConv7x7Index(3MLIB),
mllib_ImageConvKernelConvert(3MLIB), mllib_ImageConvMxN(3MLIB),
mllib_ImageConvMxN_Fp(3MLIB), mllib_ImageConvMxNIndex(3MLIB),
mllib_ImageConvolveMxN(3MLIB), mllib_ImageConvolveMxN_Fp(3MLIB),
mllib_ImageSConv3x3(3MLIB), mllib_ImageSConv3x3_Fp(3MLIB),
mllib_ImageSConv5x5(3MLIB), mllib_ImageSConv5x5_Fp(3MLIB),
mllib_ImageSConv7x7(3MLIB), mllib_ImageSConv7x7_Fp(3MLIB),
mllib_ImageSConvKernelConvert(3MLIB), attributes(5)

NAME	mllib_ImageConv4x4_Fp – 4x4 convolution
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConv4x4_Fp(mllib_image *dst, const mllib_image *src, const mllib_d64 *kernel, mllib_s32 cmask, mllib_edge edge);</pre>
DESCRIPTION	<p>The <code>mllib_ImageConv4x4_Fp()</code> function performs a floating-point 4x4 convolution on the source image by using the user-supplied kernel.</p> <p>The input image and the output image must have the same image type and have the same number of channels. The unselected channels in the output image are not overwritten. For single-channel images, the channel mask is ignored.</p> <p>For this convolution, the key element of the convolution kernel is located at (1, 1) of the kernel matrix.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sum_{p=-dm}^{m-1-dm} \sum_{q=-dn}^{n-1-dn} src[x+p][y+q][i] * k[p][q]$ <p>where $m = 4$, $n = 4$, $dm = (m - 1) / 2 = 1$, $dn = (n - 1) / 2 = 1$.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>kernel</i> Pointer to the convolution kernel, in row major order.</p> <p><i>cmask</i> Channel mask to indicate the channels to be convolved, each bit of which represents a channel in the image. The channels corresponding to bits with a value of 1 are those to be processed. For a single-channel image, the channel mask is ignored.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SRC_EXTEND</pre>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:

mllib_ImageConv4x4_Fp(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

mllib_ImageConv2x2(3MLIB), mllib_ImageConv2x2_Fp(3MLIB),
mllib_ImageConv2x2Index(3MLIB), mllib_ImageConv3x3(3MLIB),
mllib_ImageConv3x3_Fp(3MLIB), mllib_ImageConv3x3Index(3MLIB),
mllib_ImageConv4x4(3MLIB), mllib_ImageConv4x4Index(3MLIB),
mllib_ImageConv5x5(3MLIB), mllib_ImageConv5x5_Fp(3MLIB),
mllib_ImageConv5x5Index(3MLIB), mllib_ImageConv7x7(3MLIB),
mllib_ImageConv7x7_Fp(3MLIB), mllib_ImageConv7x7Index(3MLIB),
mllib_ImageConvKernelConvert(3MLIB), mllib_ImageConvMxN(3MLIB),
mllib_ImageConvMxN_Fp(3MLIB), mllib_ImageConvMxNIndex(3MLIB),
mllib_ImageConvolveMxN(3MLIB), mllib_ImageConvolveMxN_Fp(3MLIB),
mllib_ImageSConv3x3(3MLIB), mllib_ImageSConv3x3_Fp(3MLIB),
mllib_ImageSConv5x5(3MLIB), mllib_ImageSConv5x5_Fp(3MLIB),
mllib_ImageSConv7x7(3MLIB), mllib_ImageSConv7x7_Fp(3MLIB),
mllib_ImageSConvKernelConvert(3MLIB), attributes(5)

NAME	mllib_ImageConv4x4Index – 4x4 convolution on a color indexed image
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConv4x4Index(mllib_image *dst, const mllib_image *src, const mllib_s32 *kernel, mllib_s32 scale, mllib_edge edge, const void *colormap);</pre>
DESCRIPTION	<p>The <code>mllib_ImageConv4x4Index()</code> function performs a 4x4 convolution on the color indexed source image by using the user-supplied kernel.</p> <p>The input and output images must have the same image type and size.</p> <p>For this convolution, the key element of the convolution kernel is located at (1, 1) of the kernel matrix.</p> <p>This function performs the convolution on color indexed image. The input image and the output image must be single channel images. The image type must be <code>MLIB_BYTE</code> or <code>MLIB_SHORT</code>.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sum_{p=-dm}^{m-1-dm} \sum_{q=-dn}^{n-1-dn} src[x+p][y+q][i] * k[p][q] * 2^{**}(-scale)$ <p>where $m = 4$, $n = 4$, $dm = (m - 1) / 2 = 1$, $dn = (n - 1) / 2 = 1$.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>kernel</i> Pointer to the convolution kernel, in row major order.</p> <p><i>scale</i> Scaling factor.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SRC_EXTEND</pre> <p><i>colormap</i> Internal data structure for inverse color mapping. This data structure is generated by the <code>mllib_ImageColorTrue2IndexInit()</code> function.</p>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .

mllib_ImageConv4x4Index(3MLIB)

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageConv2x2(3MLIB), mllib_ImageConv2x2_Fp(3MLIB),
mllib_ImageConv2x2Index(3MLIB), mllib_ImageConv3x3(3MLIB),
mllib_ImageConv3x3_Fp(3MLIB), mllib_ImageConv3x3Index(3MLIB),
mllib_ImageConv4x4(3MLIB), mllib_ImageConv4x4_Fp(3MLIB),
mllib_ImageConv5x5(3MLIB), mllib_ImageConv5x5_Fp(3MLIB),
mllib_ImageConv5x5Index(3MLIB), mllib_ImageConv7x7(3MLIB),
mllib_ImageConv7x7_Fp(3MLIB), mllib_ImageConv7x7Index(3MLIB),
mllib_ImageConvKernelConvert(3MLIB), mllib_ImageConvMxN(3MLIB),
mllib_ImageConvMxN_Fp(3MLIB), mllib_ImageConvMxNIndex(3MLIB),
mllib_ImageConvolveMxN(3MLIB), mllib_ImageConvolveMxN_Fp(3MLIB),
mllib_ImageSConv3x3(3MLIB), mllib_ImageSConv3x3_Fp(3MLIB),
mllib_ImageSConv5x5(3MLIB), mllib_ImageSConv5x5_Fp(3MLIB),
mllib_ImageSConv7x7(3MLIB), mllib_ImageSConv7x7_Fp(3MLIB),
mllib_ImageSConvKernelConvert(3MLIB), attributes(5)

NAME	mllib_ImageConv5x5 – 5x5 convolution
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConv5x5(mllib_image *dst, const mllib_image *src, const mllib_s32 *kernel, mllib_s32 scale, mllib_s32 cmask, mllib_edge edge);</pre>
DESCRIPTION	<p>The <code>mllib_ImageConv5x5()</code> function performs a 5x5 convolution on the source image by using the user-supplied kernel.</p> <p>The input image and the output image must have the same image type and have the same number of channels. The unselected channels in the output image are not overwritten. For single-channel images, the channel mask is ignored.</p> <p>For this convolution, the key element of the convolution kernel is located at the center of the kernel matrix.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sum_{p=-dm}^{m-1-dm} \sum_{q=-dn}^{n-1-dn} src[x+p][y+q][i] * k[p][q] * 2^{**}(-scale)$ <p>where $m = 5$, $n = 5$, $dm = (m - 1) / 2 = 2$, $dn = (n - 1) / 2 = 2$.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>kernel</i> Pointer to the convolution kernel, in row major order.</p> <p><i>scale</i> Scaling factor.</p> <p><i>cmask</i> Channel mask to indicate the channels to be convolved, each bit of which represents a channel in the image. The channels corresponding to bits with a value of 1 are those to be processed. For a single-channel image, the channel mask is ignored.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SRC_EXTEND</pre>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .

mllib_ImageConv5x5(3MLIB)

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageConv2x2(3MLIB), mllib_ImageConv2x2_Fp(3MLIB),
mllib_ImageConv2x2Index(3MLIB), mllib_ImageConv3x3(3MLIB),
mllib_ImageConv3x3_Fp(3MLIB), mllib_ImageConv3x3Index(3MLIB),
mllib_ImageConv4x4(3MLIB), mllib_ImageConv4x4_Fp(3MLIB),
mllib_ImageConv4x4Index(3MLIB), mllib_ImageConv5x5_Fp(3MLIB),
mllib_ImageConv5x5Index(3MLIB), mllib_ImageConv7x7(3MLIB),
mllib_ImageConv7x7_Fp(3MLIB), mllib_ImageConv7x7Index(3MLIB),
mllib_ImageConvKernelConvert(3MLIB), mllib_ImageConvMxN(3MLIB),
mllib_ImageConvMxN_Fp(3MLIB), mllib_ImageConvMxNIndex(3MLIB),
mllib_ImageConvolveMxN(3MLIB), mllib_ImageConvolveMxN_Fp(3MLIB),
mllib_ImageSConv3x3(3MLIB), mllib_ImageSConv3x3_Fp(3MLIB),
mllib_ImageSConv5x5(3MLIB), mllib_ImageSConv5x5_Fp(3MLIB),
mllib_ImageSConv7x7(3MLIB), mllib_ImageSConv7x7_Fp(3MLIB),
mllib_ImageSConvKernelConvert(3MLIB), attributes(5)

NAME	mllib_ImageConv5x5_Fp – 5x5 convolution
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConv5x5_Fp(mllib_image *dst, const mllib_image *src, const mllib_d64 *kernel, mllib_s32 cmask, mllib_edge edge);</pre>
DESCRIPTION	<p>The <code>mllib_ImageConv5x5_Fp()</code> function performs a floating-point 5x5 convolution on the source image by using the user-supplied kernel.</p> <p>The input image and the output image must have the same image type and have the same number of channels. The unselected channels in the output image are not overwritten. For single-channel images, the channel mask is ignored.</p> <p>For this convolution, the key element of the convolution kernel is located at the center of the kernel matrix.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sum_{p=-dm}^{m-1-dm} \sum_{q=-dn}^{n-1-dn} src[x+p][y+q][i] * k[p][q]$ <p>where $m = 5$, $n = 5$, $dm = (m - 1) / 2 = 2$, $dn = (n - 1) / 2 = 2$.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>kernel</i> Pointer to the convolution kernel, in row major order.</p> <p><i>cmask</i> Channel mask to indicate the channels to be convolved, each bit of which represents a channel in the image. The channels corresponding to bits with a value of 1 are those to be processed. For a single-channel image, the channel mask is ignored.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SRC_EXTEND</pre>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:

mllib_ImageConv5x5_Fp(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

mllib_ImageConv2x2(3MLIB), mllib_ImageConv2x2_Fp(3MLIB),
mllib_ImageConv2x2Index(3MLIB), mllib_ImageConv3x3(3MLIB),
mllib_ImageConv3x3_Fp(3MLIB), mllib_ImageConv3x3Index(3MLIB),
mllib_ImageConv4x4(3MLIB), mllib_ImageConv4x4_Fp(3MLIB),
mllib_ImageConv4x4Index(3MLIB), mllib_ImageConv5x5(3MLIB),
mllib_ImageConv5x5Index(3MLIB), mllib_ImageConv7x7(3MLIB),
mllib_ImageConv7x7_Fp(3MLIB), mllib_ImageConv7x7Index(3MLIB),
mllib_ImageConvKernelConvert(3MLIB), mllib_ImageConvMxN(3MLIB),
mllib_ImageConvMxN_Fp(3MLIB), mllib_ImageConvMxNIndex(3MLIB),
mllib_ImageConvolveMxN(3MLIB), mllib_ImageConvolveMxN_Fp(3MLIB),
mllib_ImageSConv3x3(3MLIB), mllib_ImageSConv3x3_Fp(3MLIB),
mllib_ImageSConv5x5(3MLIB), mllib_ImageSConv5x5_Fp(3MLIB),
mllib_ImageSConv7x7(3MLIB), mllib_ImageSConv7x7_Fp(3MLIB),
mllib_ImageSConvKernelConvert(3MLIB), attributes(5)

NAME	mllib_ImageConv5x5Index – 5x5 convolution on a color indexed image
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConv5x5Index(mllib_image *dst, const mllib_image *src, const mllib_s32 *kernel, mllib_s32 scale, mllib_edge edge, const void *colormap);</pre>
DESCRIPTION	<p>The <code>mllib_ImageConv5x5Index()</code> function performs a 5x5 convolution the color indexed source image by using the user-supplied kernel. The source and destination images must be single-channel images.</p> <p>The input and output images must have the same image type and size.</p> <p>For this convolution, the key element of the convolution kernel is located at the center of the kernel matrix.</p> <p>This function performs the convolution on color indexed image. The input image and the output image must be single channel images. The image type must be <code>MLIB_BYTE</code> or <code>MLIB_SHORT</code>.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sum_{p=-dm}^{m-1-dm} \sum_{q=-dn}^{n-1-dn} src[x+p][y+q][i] * k[p][q] * 2^{*(-scale)}$ <p>where $m = 5$, $n = 5$, $dm = (m - 1) / 2 = 2$, $dn = (n - 1) / 2 = 2$.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>kernel</i> Pointer to the convolution kernel, in row major order.</p> <p><i>scale</i> Scaling factor.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SRC_EXTEND</pre> <p><i>colormap</i> Internal data structure for inverse color mapping. This data structure is generated by the <code>mllib_ImageColorTrue2IndexInit()</code> function.</p>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .

mllib_ImageConv5x5Index(3MLIB)

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageConv2x2(3MLIB), mllib_ImageConv2x2_Fp(3MLIB), mllib_ImageConv2x2Index(3MLIB), mllib_ImageConv3x3(3MLIB), mllib_ImageConv3x3_Fp(3MLIB), mllib_ImageConv3x3Index(3MLIB), mllib_ImageConv4x4(3MLIB), mllib_ImageConv4x4_Fp(3MLIB), mllib_ImageConv4x4Index(3MLIB), mllib_ImageConv5x5(3MLIB), mllib_ImageConv5x5_Fp(3MLIB), mllib_ImageConv7x7(3MLIB), mllib_ImageConv7x7_Fp(3MLIB), mllib_ImageConv7x7Index(3MLIB), mllib_ImageConvKernelConvert(3MLIB), mllib_ImageConvMxN(3MLIB), mllib_ImageConvMxN_Fp(3MLIB), mllib_ImageConvMxNIndex(3MLIB), mllib_ImageConvolveMxN(3MLIB), mllib_ImageConvolveMxN_Fp(3MLIB), mllib_ImageSConv3x3(3MLIB), mllib_ImageSConv3x3_Fp(3MLIB), mllib_ImageSConv5x5(3MLIB), mllib_ImageSConv5x5_Fp(3MLIB), mllib_ImageSConv7x7(3MLIB), mllib_ImageSConv7x7_Fp(3MLIB), mllib_ImageSConvKernelConvert(3MLIB), attributes(5)

NAME	mllib_ImageConv7x7 – 7x7 convolution
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConv7x7(mllib_image *dst, const mllib_image *src, const mllib_s32 *kernel, mllib_s32 scale, mllib_s32 cmask, mllib_edge edge);</pre>
DESCRIPTION	<p>The <code>mllib_ImageConv7x7()</code> function performs a 7x7 convolution on the source image by using the user-supplied kernel.</p> <p>The input image and the output image must have the same image type and have the same number of channels. The unselected channels in the output image are not overwritten. For single-channel images, the channel mask is ignored.</p> <p>For this convolution, the key element of the convolution kernel is located at the center of the kernel matrix.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sum_{p=-dm}^{m-1-dm} \sum_{q=-dn}^{n-1-dn} src[x+p][y+q][i] * k[p][q] * 2^{**}(-scale)$ <p>where $m = 7$, $n = 7$, $dm = (m - 1) / 2 = 3$, $dn = (n - 1) / 2 = 3$.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>kernel</i> Pointer to the convolution kernel, in row major order.</p> <p><i>scale</i> Scaling factor.</p> <p><i>cmask</i> Channel mask to indicate the channels to be convolved, each bit of which represents a channel in the image. The channels corresponding to bits with a value of 1 are those to be processed. For a single-channel image, the channel mask is ignored.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SRC_EXTEND</pre>
RETURN VALUES	<p>The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code>.</p>

mllib_ImageConv7x7(3MLIB)

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_ImageConv2x2(3MLIB), mllib_ImageConv2x2_Fp(3MLIB),
mllib_ImageConv2x2Index(3MLIB), mllib_ImageConv3x3(3MLIB),
mllib_ImageConv3x3_Fp(3MLIB), mllib_ImageConv3x3Index(3MLIB),
mllib_ImageConv4x4(3MLIB), mllib_ImageConv4x4_Fp(3MLIB),
mllib_ImageConv4x4Index(3MLIB), mllib_ImageConv5x5(3MLIB),
mllib_ImageConv5x5_Fp(3MLIB), mllib_ImageConv5x5Index(3MLIB),
mllib_ImageConv7x7_Fp(3MLIB), mllib_ImageConv7x7Index(3MLIB),
mllib_ImageConvKernelConvert(3MLIB), mllib_ImageConvMxN(3MLIB),
mllib_ImageConvMxN_Fp(3MLIB), mllib_ImageConvMxNIndex(3MLIB),
mllib_ImageConvolveMxN(3MLIB), mllib_ImageConvolveMxN_Fp(3MLIB),
mllib_ImageSConv3x3(3MLIB), mllib_ImageSConv3x3_Fp(3MLIB),
mllib_ImageSConv5x5(3MLIB), mllib_ImageSConv5x5_Fp(3MLIB),
mllib_ImageSConv7x7(3MLIB), mllib_ImageSConv7x7_Fp(3MLIB),
mllib_ImageSConvKernelConvert(3MLIB), attributes(5)

NAME	mllib_ImageConv7x7_Fp – 7x7 convolution
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConv7x7_Fp(mllib_image *dst, const mllib_image *src, const mllib_d64 *kernel, mllib_s32 cmask, mllib_edge edge);</pre>
DESCRIPTION	<p>The <code>mllib_ImageConv7x7_Fp()</code> function performs a floating-point 7x7 convolution on the source image by using the user-supplied kernel.</p> <p>The input image and the output image must have the same image type and have the same number of channels. The unselected channels in the output image are not overwritten. For single-channel images, the channel mask is ignored.</p> <p>For this convolution, the key element of the convolution kernel is located at the center of the kernel matrix.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sum_{p=-dm}^{m-1-dm} \sum_{q=-dn}^{n-1-dn} src[x+p][y+q][i] * k[p][q]$ <p>where $m = 7$, $n = 7$, $dm = (m - 1) / 2 = 3$, $dn = (n - 1) / 2 = 3$.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>kernel</i> Pointer to the convolution kernel, in row major order.</p> <p><i>cmask</i> Channel mask to indicate the channels to be convolved, each bit of which represents a channel in the image. The channels corresponding to bits with a value of 1 are those to be processed. For a single-channel image, the channel mask is ignored.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SRC_EXTEND</pre>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:

mllib_ImageConv7x7_Fp(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

mllib_ImageConv2x2(3MLIB), mllib_ImageConv2x2_Fp(3MLIB),
mllib_ImageConv2x2Index(3MLIB), mllib_ImageConv3x3(3MLIB),
mllib_ImageConv3x3_Fp(3MLIB), mllib_ImageConv3x3Index(3MLIB),
mllib_ImageConv4x4(3MLIB), mllib_ImageConv4x4_Fp(3MLIB),
mllib_ImageConv4x4Index(3MLIB), mllib_ImageConv5x5(3MLIB),
mllib_ImageConv5x5_Fp(3MLIB), mllib_ImageConv5x5Index(3MLIB),
mllib_ImageConv7x7(3MLIB), mllib_ImageConv7x7Index(3MLIB),
mllib_ImageConvKernelConvert(3MLIB), mllib_ImageConvMxN(3MLIB),
mllib_ImageConvMxN_Fp(3MLIB), mllib_ImageConvMxNIndex(3MLIB),
mllib_ImageConvolveMxN(3MLIB), mllib_ImageConvolveMxN_Fp(3MLIB),
mllib_ImageSConv3x3(3MLIB), mllib_ImageSConv3x3_Fp(3MLIB),
mllib_ImageSConv5x5(3MLIB), mllib_ImageSConv5x5_Fp(3MLIB),
mllib_ImageSConv7x7(3MLIB), mllib_ImageSConv7x7_Fp(3MLIB),
mllib_ImageSConvKernelConvert(3MLIB), attributes(5)

NAME	mllib_ImageConv7x7Index – 7x7 convolution on a color indexed image
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConv7x7Index(mllib_image *dst, const mllib_image *src, const mllib_s32 *kernel, mllib_s32 scale, mllib_edge edge, const void *colormap);</pre>
DESCRIPTION	<p>The <code>mllib_ImageConv7x7Index()</code> function performs a 7x7 convolution the color indexed source image by using the user-supplied kernel. The source and destination images must be single-channel images.</p> <p>The input and output images must have the same image type and size.</p> <p>For this convolution, the key element of the convolution kernel is located at the center of the kernel matrix.</p> <p>This function performs the convolution on color indexed image. The input image and the output image must be single channel images. The image type must be <code>MLIB_BYTE</code> or <code>MLIB_SHORT</code>.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sum_{p=-dm}^{m-1-dm} \sum_{q=-dn}^{n-1-dn} src[x+p][y+q][i] * k[p][q] * 2^{*(-scale)}$ <p>where $m = 7$, $n = 7$, $dm = (m - 1) / 2 = 3$, $dn = (n - 1) / 2 = 3$.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>kernel</i> Pointer to the convolution kernel, in row major order.</p> <p><i>scale</i> Scaling factor.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SRC_EXTEND</pre> <p><i>colormap</i> Internal data structure for inverse color mapping. This data structure is generated by the <code>mllib_ImageColorTrue2IndexInit()</code> function.</p>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .

mllib_ImageConv7x7Index(3MLIB)

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageConv2x2(3MLIB), mllib_ImageConv2x2_Fp(3MLIB),
mllib_ImageConv2x2Index(3MLIB), mllib_ImageConv3x3(3MLIB),
mllib_ImageConv3x3_Fp(3MLIB), mllib_ImageConv3x3Index(3MLIB),
mllib_ImageConv4x4(3MLIB), mllib_ImageConv4x4_Fp(3MLIB),
mllib_ImageConv4x4Index(3MLIB), mllib_ImageConv5x5(3MLIB),
mllib_ImageConv5x5_Fp(3MLIB), mllib_ImageConv5x5Index(3MLIB),
mllib_ImageConv7x7(3MLIB), mllib_ImageConv7x7_Fp(3MLIB),
mllib_ImageConvKernelConvert(3MLIB), mllib_ImageConvMxN(3MLIB),
mllib_ImageConvMxN_Fp(3MLIB), mllib_ImageConvMxNIndex(3MLIB),
mllib_ImageConvolveMxN(3MLIB), mllib_ImageConvolveMxN_Fp(3MLIB),
mllib_ImageSConv3x3(3MLIB), mllib_ImageSConv3x3_Fp(3MLIB),
mllib_ImageSConv5x5(3MLIB), mllib_ImageSConv5x5_Fp(3MLIB),
mllib_ImageSConv7x7(3MLIB), mllib_ImageSConv7x7_Fp(3MLIB),
mllib_ImageSConvKernelConvert(3MLIB), attributes(5)

mllib_ImageConvKernelConvert(3MLIB)

NAME mllib_ImageConvKernelConvert – convolution kernel conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageConvKernelConvert(mllib_s32 *ikernel, mllib_s32
    *iscale, const mllib_d64 *fkernel, mllib_s32 m, mllib_s32 n,
    mllib_type type);
```

DESCRIPTION The `mllib_ImageConvKernelConvert()` function converts a floating-point convolution kernel to an integer kernel with its scaling factor suitable to be used in convolution functions.

PARAMETERS The function takes the following arguments:

<i>ikernel</i>	Pointer to integer convolution kernel, in row major order.
<i>iscale</i>	Pointer to scaling factor of the integer convolution kernel.
<i>fkernel</i>	Pointer to floating-point convolution kernel, in row major order.
<i>m</i>	Width of the convolution kernel. $m \geq 1$.
<i>n</i>	Height of the convolution kernel. $n \geq 1$.
<i>type</i>	The image type. It can be one of the following:

```

MLIB_BIT
MLIB_BYTE
MLIB_SHORT
MLIB_USHORT
MLIB_INT
```

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageConv2x2(3MLIB)`, `mllib_ImageConv2x2_Fp(3MLIB)`, `mllib_ImageConv2x2Index(3MLIB)`, `mllib_ImageConv3x3(3MLIB)`, `mllib_ImageConv3x3_Fp(3MLIB)`, `mllib_ImageConv3x3Index(3MLIB)`, `mllib_ImageConv4x4(3MLIB)`, `mllib_ImageConv4x4_Fp(3MLIB)`, `mllib_ImageConv4x4Index(3MLIB)`, `mllib_ImageConv5x5(3MLIB)`, `mllib_ImageConv5x5_Fp(3MLIB)`, `mllib_ImageConv5x5Index(3MLIB)`, `mllib_ImageConv7x7(3MLIB)`, `mllib_ImageConv7x7_Fp(3MLIB)`, `mllib_ImageConvMxN(3MLIB)`, `mllib_ImageConvMxNIndex(3MLIB)`, `mllib_ImageConvMxN_Fp(3MLIB)`, `mllib_ImageConvolveMxN(3MLIB)`, `mllib_ImageConvolveMxN_Fp(3MLIB)`,

`mlib_ImageConvKernelConvert(3MLIB)`

```
mlib_ImageSConv3x3(3MLIB), mlib_ImageSConv3x3_Fp(3MLIB),  
mlib_ImageSConv5x5(3MLIB), mlib_ImageSConv5x5_Fp(3MLIB),  
mlib_ImageSConv7x7(3MLIB), mlib_ImageSConv7x7_Fp(3MLIB),  
mlib_ImageSConvKernelConvert(3MLIB), attributes(5)
```

NAME	mlib_ImageConvMxN – MxN convolution
SYNOPSIS	<pre>cc [flag...] file... -lmlib [library...] #include <mlib.h> mlib_status mlib_ImageConvMxN(mlib_image *dst, const mlib_image *src, const mlib_s32 *kernel, mlib_s32 m, mlib_s32 n, mlib_s32 dm, mlib_s32 dn, mlib_s32 scale, mlib_s32 cmask, mlib_edge edge);</pre>
DESCRIPTION	<p>The <code>mlib_ImageConvMxN()</code> function performs a MxN convolution on the source image by using the user-supplied kernel.</p> <p>The input image and the output image must have the same image type and have the same number of channels. The unselected channels in the output image are not overwritten. For single-channel images, the channel mask is ignored.</p> <p>For this convolution, the key element of the convolution kernel is located at (dm, dn) of the kernel matrix.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sum_{p=-dm}^{m-1-dm} \sum_{q=-dn}^{n-1-dn} src[x+p][y+q][i] * k[p][q] * 2^{**}(-scale)$ <p>where $m \geq 1$, $n \geq 1$, $0 \leq dm < m$, $0 \leq dn < n$.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>kernel</i> Pointer to the convolution kernel, in row major order.</p> <p><i>m</i> Width of the convolution kernel. $m \geq 1$.</p> <p><i>n</i> Height of the convolution kernel. $n \geq 1$.</p> <p><i>dm</i> X coordinate of the key element in the convolution kernel. $0 \leq dm < m$.</p> <p><i>dn</i> Y coordinate of the key element in the convolution kernel. $0 \leq dn < n$.</p> <p><i>scale</i> Scaling factor.</p> <p><i>cmask</i> Channel mask to indicate the channels to be convolved, each bit of which represents a channel in the image. The channels corresponding to bits with a value of 1 are those to be processed. For a single-channel image, the channel mask is ignored.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p>

mllib_ImageConvMxN(3MLIB)

MLIB_EDGE_DST_NO_WRITE
MLIB_EDGE_DST_FILL_ZERO
MLIB_EDGE_DST_COPY_SRC
MLIB_EDGE_SRC_EXTEND

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageConv2x2(3MLIB), mllib_ImageConv2x2_Fp(3MLIB), mllib_ImageConv2x2Index(3MLIB), mllib_ImageConv3x3(3MLIB), mllib_ImageConv3x3_Fp(3MLIB), mllib_ImageConv3x3Index(3MLIB), mllib_ImageConv4x4(3MLIB), mllib_ImageConv4x4_Fp(3MLIB), mllib_ImageConv4x4Index(3MLIB), mllib_ImageConv5x5(3MLIB), mllib_ImageConv5x5_Fp(3MLIB), mllib_ImageConv5x5Index(3MLIB), mllib_ImageConv7x7(3MLIB), mllib_ImageConv7x7_Fp(3MLIB), mllib_ImageConv7x7Index(3MLIB), mllib_ImageConvKernelConvert(3MLIB), mllib_ImageConvMxN_Fp(3MLIB), mllib_ImageConvMxNIndex(3MLIB), mllib_ImageConvolveMxN(3MLIB), mllib_ImageConvolveMxN_Fp(3MLIB), mllib_ImageSConv3x3(3MLIB), mllib_ImageSConv3x3_Fp(3MLIB), mllib_ImageSConv5x5(3MLIB), mllib_ImageSConv5x5_Fp(3MLIB), mllib_ImageSConv7x7(3MLIB), mllib_ImageSConv7x7_Fp(3MLIB), mllib_ImageSConvKernelConvert(3MLIB), attributes(5)

NAME	mllib_ImageConvMxN_Fp – MxN convolution
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConvMxN_Fp(mllib_image *dst, const mllib_image *src, const mllib_d64 *kernel, mllib_s32 m, mllib_s32 n, mllib_s32 dm, mllib_s32 dn, mllib_s32 cmask, mllib_edge edge);</pre>
DESCRIPTION	<p>The <code>mllib_ImageConvMxN_Fp()</code> function performs a MxN convolution on the source image by using the user-supplied kernel.</p> <p>The input image and the output image must have the same image type and have the same number of channels. The unselected channels in the output image are not overwritten. For single-channel images, the channel mask is ignored.</p> <p>For this convolution, the key element of the convolution kernel is located at (dm, dn) of the kernel matrix.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sum_{p=-dm}^{m-1-dm} \sum_{q=-dn}^{n-1-dn} src[x+p][y+q][i] * k[p][q]$ <p>where $m \geq 1$, $n \geq 1$, $0 \leq dm < m$, $0 \leq dn < n$.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>kernel</i> Pointer to the convolution kernel, in row major order.</p> <p><i>m</i> Width of the convolution kernel. $m \geq 1$.</p> <p><i>n</i> Height of the convolution kernel. $n \geq 1$.</p> <p><i>dm</i> X coordinate of the key element in the convolution kernel. $0 \leq dm < m$.</p> <p><i>dn</i> Y coordinate of the key element in the convolution kernel. $0 \leq dn < n$.</p> <p><i>cmask</i> Channel mask to indicate the channels to be convolved, each bit of which represents a channel in the image. The channels corresponding to bits with a value of 1 are those to be processed. For a single-channel image, the channel mask is ignored.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SRC_EXTEND</pre>

`mllib_ImageConvMxN_Fp(3MLIB)`

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageConv2x2(3MLIB)`, `mllib_ImageConv2x2_Fp(3MLIB)`, `mllib_ImageConv2x2Index(3MLIB)`, `mllib_ImageConv3x3(3MLIB)`, `mllib_ImageConv3x3_Fp(3MLIB)`, `mllib_ImageConv3x3Index(3MLIB)`, `mllib_ImageConv4x4(3MLIB)`, `mllib_ImageConv4x4_Fp(3MLIB)`, `mllib_ImageConv4x4Index(3MLIB)`, `mllib_ImageConv5x5(3MLIB)`, `mllib_ImageConv5x5_Fp(3MLIB)`, `mllib_ImageConv5x5Index(3MLIB)`, `mllib_ImageConv7x7(3MLIB)`, `mllib_ImageConv7x7_Fp(3MLIB)`, `mllib_ImageConv7x7Index(3MLIB)`, `mllib_ImageConvKernelConvert(3MLIB)`, `mllib_ImageConvMxN(3MLIB)`, `mllib_ImageConvMxNIndex(3MLIB)`, `mllib_ImageConvolveMxN(3MLIB)`, `mllib_ImageConvolveMxN_Fp(3MLIB)`, `mllib_ImageSConv3x3(3MLIB)`, `mllib_ImageSConv3x3_Fp(3MLIB)`, `mllib_ImageSConv5x5(3MLIB)`, `mllib_ImageSConv5x5_Fp(3MLIB)`, `mllib_ImageSConv7x7(3MLIB)`, `mllib_ImageSConv7x7_Fp(3MLIB)`, `mllib_ImageSConvKernelConvert(3MLIB)`, `attributes(5)`

NAME	mllib_ImageConvMxNIndex – MxN convolution on a color indexed image
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConvMxNIndex(mllib_image *dst, const mllib_image *src, const mllib_s32 *kernel, mllib_s32 m, mllib_s32 n, mllib_s32 dm, mllib_s32 dn, mllib_s32 scale, mllib_edge edge, const void *colormap);</pre>
DESCRIPTION	<p>The <code>mllib_ImageConvMxNIndex()</code> function performs a MxN convolution on the color indexed source image by using the user-supplied kernel.</p> <p>The input and output images must have the same image type and size.</p> <p>For this convolution, the key element of the convolution kernel is located at (dm, dn) of the kernel matrix.</p> <p>This function performs the convolution on a color indexed image. The input image and the output image must be single-channel images. The image type must be <code>MLIB_BYTE</code> or <code>MLIB_SHORT</code>.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sum_{p=-dm}^{m-1-dm} \sum_{q=-dn}^{n-1-dn} src[x+p][y+q][i] * k[p][q] * 2^{**(-scale)}$ <p>where $m > 1$, $n > 1$, $0 \leq dm < m$, $0 \leq dn < n$.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>kernel</i> Pointer to the convolution kernel, in row major order.</p> <p><i>m</i> Width of the convolution kernel. $m > 1$.</p> <p><i>n</i> Height of the convolution kernel. $n > 1$.</p> <p><i>dm</i> X coordinate of the key element in the convolution kernel. $0 \leq dm < m$.</p> <p><i>dn</i> Y coordinate of the key element in the convolution kernel. $0 \leq dn < n$.</p> <p><i>scale</i> Scaling factor.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SRC_EXTEND</pre>

mllib_ImageConvMxNIndex(3MLIB)

colormap Internal data structure for inverse color mapping. This data structure is generated by the `mllib_ImageColorTrue2IndexInit()` function.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageConv2x2(3MLIB)`, `mllib_ImageConv2x2_Fp(3MLIB)`, `mllib_ImageConv2x2Index(3MLIB)`, `mllib_ImageConv3x3(3MLIB)`, `mllib_ImageConv3x3_Fp(3MLIB)`, `mllib_ImageConv3x3Index(3MLIB)`, `mllib_ImageConv4x4(3MLIB)`, `mllib_ImageConv4x4_Fp(3MLIB)`, `mllib_ImageConv4x4Index(3MLIB)`, `mllib_ImageConv5x5(3MLIB)`, `mllib_ImageConv5x5_Fp(3MLIB)`, `mllib_ImageConv5x5Index(3MLIB)`, `mllib_ImageConv7x7(3MLIB)`, `mllib_ImageConv7x7_Fp(3MLIB)`, `mllib_ImageConv7x7Index(3MLIB)`, `mllib_ImageConvKernelConvert(3MLIB)`, `mllib_ImageConvMxN(3MLIB)`, `mllib_ImageConvMxN_Fp(3MLIB)`, `mllib_ImageConvolveMxN(3MLIB)`, `mllib_ImageConvolveMxN_Fp(3MLIB)`, `mllib_ImageSConv3x3(3MLIB)`, `mllib_ImageSConv3x3_Fp(3MLIB)`, `mllib_ImageSConv5x5(3MLIB)`, `mllib_ImageSConv5x5_Fp(3MLIB)`, `mllib_ImageSConv7x7(3MLIB)`, `mllib_ImageSConv7x7_Fp(3MLIB)`, `mllib_ImageSConvKernelConvert(3MLIB)`, `attributes(5)`

NAME	mllib_ImageConvolveMxN – MxN convolution, with kernel analysis for taking advantage of special cases
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConvolveMxN(mllib_image *dst, const mllib_image *src, const mllib_d64 *kernel, mllib_s32 m, mllib_s32 n, mllib_s32 dm, mllib_s32 dn, mllib_s32 cmask, mllib_edge edge);</pre>
DESCRIPTION	<p>The <code>mllib_ImageConvolveMxN()</code> function analyzes the convolution kernel, converts the floating-point kernel to an integer kernel, then performs a MxN convolution on the source image by calling either one of the functions like <code>mllib_ImageSConv3x3()</code>, <code>mllib_ImageConv3x3()</code>, and etc. in special cases or <code>mllib_ImageConvMxN()</code> in other cases.</p> <p>The input image and the output image must have the same image type and have the same number of channels. The unselected channels in the output image are not overwritten. For single-channel images, the channel mask is ignored.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sum_{p=-dm}^{m-1-dm} \sum_{q=-dn}^{n-1-dn} src[x+p][y+q][i] * k[p][q]$ <p>where $m \geq 1$, $n \geq 1$, $0 \leq dm < m$, $0 \leq dn < n$.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>kernel</i> Pointer to the convolution kernel, in row major order.</p> <p><i>m</i> Width of the convolution kernel. $m \geq 1$.</p> <p><i>n</i> Height of the convolution kernel. $n \geq 1$.</p> <p><i>dm</i> X coordinate of the key element in the convolution kernel. $0 \leq dm < m$.</p> <p><i>dn</i> Y coordinate of the key element in the convolution kernel. $0 \leq dn < n$.</p> <p><i>cmask</i> Channel mask to indicate the channels to be convolved, each bit of which represents a channel in the image. The channels corresponding to 1 bits are those to be processed. For a single-channel image, the channel mask is ignored.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p>

mllib_ImageConvolveMxN(3MLIB)

MLIB_EDGE_DST_NO_WRITE
MLIB_EDGE_DST_FILL_ZERO
MLIB_EDGE_DST_COPY_SRC
MLIB_EDGE_SRC_EXTEND

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageConv2x2(3MLIB), mllib_ImageConv2x2_Fp(3MLIB), mllib_ImageConv2x2Index(3MLIB), mllib_ImageConv3x3(3MLIB), mllib_ImageConv3x3_Fp(3MLIB), mllib_ImageConv3x3Index(3MLIB), mllib_ImageConv4x4(3MLIB), mllib_ImageConv4x4_Fp(3MLIB), mllib_ImageConv4x4Index(3MLIB), mllib_ImageConv5x5(3MLIB), mllib_ImageConv5x5_Fp(3MLIB), mllib_ImageConv5x5Index(3MLIB), mllib_ImageConv7x7(3MLIB), mllib_ImageConv7x7_Fp(3MLIB), mllib_ImageConv7x7Index(3MLIB), mllib_ImageConvKernelConvert(3MLIB), mllib_ImageConvMxN(3MLIB), mllib_ImageConvMxN_Fp(3MLIB), mllib_ImageConvMxNIndex(3MLIB), mllib_ImageConvolveMxN_Fp(3MLIB), mllib_ImageSConv3x3(3MLIB), mllib_ImageSConv3x3_Fp(3MLIB), mllib_ImageSConv5x5(3MLIB), mllib_ImageSConv5x5_Fp(3MLIB), mllib_ImageSConv7x7(3MLIB), mllib_ImageSConv7x7_Fp(3MLIB), mllib_ImageSConvKernelConvert(3MLIB), attributes(5)

NAME	mllib_ImageConvolveMxN_Fp – MxN convolution, with kernel analysis for taking advantage of special cases
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageConvolveMxN_Fp(mllib_image *dst, const mllib_image *src, const mllib_d64 *kernel, mllib_s32 m, mllib_s32 n, mllib_s32 dm, mllib_s32 dn, mllib_s32 cmask, mllib_edge edge);</pre>
DESCRIPTION	<p>The <code>mllib_ImageConvolveMxN_Fp()</code> function analyzes the convolution kernel, then performs a MxN convolution on the source image by calling either one of the functions like <code>mllib_ImageSConv3x3_Fp()</code>, <code>mllib_ImageConv3x3_Fp()</code>, and etc. in special cases or <code>mllib_ImageConvMxN_Fp()</code> in other cases.</p> <p>The input image and the output image must have the same image type and have the same number of channels. The unselected channels in the output image are not overwritten. For single-channel images, the channel mask is ignored.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sum_{p=-dm}^{m-1-dm} \sum_{q=-dn}^{n-1-dn} src[x+p][y+q][i] * k[p][q]$ <p>where $m \geq 1$, $n \geq 1$, $0 \leq dm < m$, $0 \leq dn < n$.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>kernel</i> Pointer to the convolution kernel, in row major order.</p> <p><i>m</i> Width of the convolution kernel. $m \geq 1$.</p> <p><i>n</i> Height of the convolution kernel. $n \geq 1$.</p> <p><i>dm</i> X coordinate of the key element in the convolution kernel. $0 \leq dm < m$.</p> <p><i>dn</i> Y coordinate of the key element in the convolution kernel. $0 \leq dn < n$.</p> <p><i>cmask</i> Channel mask to indicate the channels to be convolved, each bit of which represents a channel in the image. The channels corresponding to 1 bits are those to be processed. For a single-channel image, the channel mask is ignored.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SRC_EXTEND</pre>

`mllib_ImageConvolveMxN_Fp(3MLIB)`

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageConv2x2(3MLIB)`, `mllib_ImageConv2x2_Fp(3MLIB)`,
`mllib_ImageConv2x2Index(3MLIB)`, `mllib_ImageConv3x3(3MLIB)`,
`mllib_ImageConv3x3_Fp(3MLIB)`, `mllib_ImageConv3x3Index(3MLIB)`,
`mllib_ImageConv4x4(3MLIB)`, `mllib_ImageConv4x4_Fp(3MLIB)`,
`mllib_ImageConv4x4Index(3MLIB)`, `mllib_ImageConv5x5(3MLIB)`,
`mllib_ImageConv5x5_Fp(3MLIB)`, `mllib_ImageConv5x5Index(3MLIB)`,
`mllib_ImageConv7x7(3MLIB)`, `mllib_ImageConv7x7_Fp(3MLIB)`,
`mllib_ImageConv7x7Index(3MLIB)`, `mllib_ImageConvKernelConvert(3MLIB)`,
`mllib_ImageConvMxN(3MLIB)`, `mllib_ImageConvMxN_Fp(3MLIB)`,
`mllib_ImageConvMxNIndex(3MLIB)`, `mllib_ImageConvolveMxN(3MLIB)`,
`mllib_ImageSConv3x3(3MLIB)`, `mllib_ImageSConv3x3_Fp(3MLIB)`,
`mllib_ImageSConv5x5(3MLIB)`, `mllib_ImageSConv5x5_Fp(3MLIB)`,
`mllib_ImageSConv7x7(3MLIB)`, `mllib_ImageSConv7x7_Fp(3MLIB)`,
`mllib_ImageSConvKernelConvert(3MLIB)`, `attributes(5)`

mlib_ImageCopy(3MLIB)

NAME | mlib_ImageCopy – image copy

SYNOPSIS |

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_ImageCopy(mlib_image *dst, const mlib_image *src);
```

DESCRIPTION | The `mlib_ImageCopy()` function copies the source image to the destination image. The data type of the images can be `MLIB_BIT`, `MLIB_BYTE`, `MLIB_SHORT`, `MLIB_USHORT`, `MLIB_INT`, `MLIB_FLOAT`, or `MLIB_DOUBLE`.
It uses the following equation:
`dst[x][y][i] = src[x][y][i]`

PARAMETERS | The function takes the following arguments:
dst Pointer to destination image.
src Pointer to source image.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mlib_ImageCopyArea(3MLIB)`, `mlib_ImageCopyMask(3MLIB)`,
`mlib_ImageCopyMask_Fp(3MLIB)`, `mlib_ImageCopySubimage(3MLIB)`,
`attributes(5)`

mllib_ImageCopyArea(3MLIB)

NAME	mllib_ImageCopyArea – copy an area						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageCopyArea(mllib_image *<i>img</i>, mllib_s32 <i>x</i>, mllib_s32 <i>y</i>, mllib_s32 <i>w</i>, mllib_s32 <i>h</i>, mllib_s32 <i>dx</i>, mllib_s32 <i>dy</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageCopyArea()</code> function copies a specified rectangular area from one portion of the image to another portion of the same image. The data type of the image can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, <code>MLIB_INT</code>, <code>MLIB_FLOAT</code>, or <code>MLIB_DOUBLE</code>.</p> <p>It uses the following equation:</p> $\text{img}[x+dx+i][y+dy+j][i] = \text{img}[x+i][y+j][i]$ <p>where $i = 0, 1, \dots, w-1$; $j = 0, 1, \dots, h-1$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>img</i> Pointer to source image.</p> <p><i>x</i> X coordinate of the area origin in the source.</p> <p><i>y</i> Y coordinate of the area origin in the source.</p> <p><i>w</i> Width of the area to be copied.</p> <p><i>h</i> Height of the area to be copied.</p> <p><i>dx</i> Horizontal displacement in pixels of the area to be copied.</p> <p><i>dy</i> Vertical displacement in pixels of the area to be copied.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageCopy(3MLIB)</code> , <code>mllib_ImageCopyMask(3MLIB)</code> , <code>mllib_ImageCopyMask_Fp(3MLIB)</code> , <code>mllib_ImageCopySubimage(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageCopyMask – copy with mask

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageCopyMask(mllib_image *dst, const mllib_image
    *src, const mllib_image *mask, const mllib_s32 *thresh);
```

DESCRIPTION The mllib_ImageCopyMask() function copies one image to another image via a mask image by using it as a yes/no indicator. The data type of the images can be MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, or MLIB_INT.

It uses the following equation:

```
dst[x][y][i] = src[x][y][i]  if mask[x][y][i] ≤ thresh[i]
dst[x][y][i] = dst[x][y][i]  if mask[x][y][i] > thresh[i]
```

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

mask Pointer to mask image.

thresh Threshold for the mask image. thresh[i] contains the threshold for channel i.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageCopy(3MLIB), mllib_ImageCopyArea(3MLIB), mllib_ImageCopyMask_Fp(3MLIB), mllib_ImageCopySubimage(3MLIB), attributes(5)

mllib_ImageCopyMask_Fp(3MLIB)

NAME	mllib_ImageCopyMask_Fp – copy with mask, floating-point						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageCopyMask_Fp(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, const mllib_image *<i>mask</i>, const mllib_d64 *<i>thresh</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageCopyMask_Fp()</code> function copies one image to another image via a mask image by using it as a yes/no indicator. The data type of the images can be <code>MLIB_FLOAT</code> or <code>MLIB_DOUBLE</code>.</p> <p>It uses the following equation:</p> $\begin{aligned} \text{dst}[x][y][i] &= \text{src}[x][y][i] && \text{if } \text{mask}[x][y][i] \leq \text{thresh}[i] \\ \text{dst}[x][y][i] &= \text{dst}[x][y][i] && \text{if } \text{mask}[x][y][i] > \text{thresh}[i] \end{aligned}$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>mask</i> Pointer to mask image.</p> <p><i>thresh</i> Threshold for the mask image. <code>thresh[i]</code> contains the threshold for channel <i>i</i>.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageCopy(3MLIB)</code> , <code>mllib_ImageCopyArea(3MLIB)</code> , <code>mllib_ImageCopyMask(3MLIB)</code> , <code>mllib_ImageCopySubimage(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageCopySubimage – copy subimage

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageCopySubimage(mllib_image *dst, const
    mllib_image *src, mllib_s32 xd, mllib_s32 yd, mllib_s32 xs,
    mllib_s32 ys, mllib_s32 w, mllib_s32 h);
```

DESCRIPTION The `mllib_ImageCopySubimage()` function copies a specified rectangular area from one image to a specified area of another image. The data type of the images can be `MLIB_BIT`, `MLIB_BYTE`, `MLIB_SHORT`, `MLIB_USHORT`, `MLIB_INT`, `MLIB_FLOAT`, or `MLIB_DOUBLE`.

It uses the following equation:

$$dst[xd+i][yd+j][i] = src[xs+i][ys+j][i]$$

where $i = 0, 1, \dots, w-1$; $j = 0, 1, \dots, h-1$.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

xd X coordinate of the area origin in the destination.

yd Y coordinate of the area origin in the destination.

xs X coordinate of the area origin in the source.

ys Y coordinate of the area origin in the source.

w Width of the area to be copied.

h Height of the area to be copied.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageCopy(3MLIB)`, `mllib_ImageCopyArea(3MLIB)`, `mllib_ImageCopyMask(3MLIB)`, `mllib_ImageCopyMask_Fp(3MLIB)`, `attributes(5)`

mllib_ImageCreate(3MLIB)

NAME mllib_ImageCreate – image creation

SYNOPSIS cc [*flag...*] *file...* -lmllib [*library...*]
#include <mllib.h>

```
mllib_image *mllib_ImageCreate(mllib_type type, mllib_s32 channels,  
                               mllib_s32 width, mllib_s32 height);
```

DESCRIPTION The mllib_ImageCreate() function creates a mediaLib image data structure and allocates memory space for image data. The data type of the image can be MLIB_BIT, MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, MLIB_INT, MLIB_FLOAT, or MLIB_DOUBLE.

To obtain the best performance, it is recommended that you use this function to create a mediaLib image whenever possible, as this guarantees alignment.

PARAMETERS The function takes the following arguments:

<i>type</i>	Image data type.
<i>channels</i>	Number of channels in the image.
<i>width</i>	Width of image in pixels.
<i>height</i>	Height of image in pixels.

RETURN VALUES The function returns a pointer to the mllib_image data structure.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageCreateStruct(3MLIB), mllib_ImageCreateSubimage(3MLIB), mllib_ImageDelete(3MLIB), mllib_ImageSetPaddings(3MLIB), attributes(5)

NAME mllib_ImageCreateStruct – image structure creation

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_image *mllib_ImageCreateStruct(mllib_type type, mllib_s32
    channels, mllib_s32 width, mllib_s32 height, mllib_s32 stride, const
    void *data);
```

DESCRIPTION The mllib_ImageCreateStruct() function creates a mediaLib image data structure with parameters supplied by the user. The data type of the image can be MLIB_BIT, MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, MLIB_INT, MLIB_FLOAT, or MLIB_DOUBLE.

PARAMETERS The function takes the following arguments:

type Image data type.

channels Number of channels in the image.

width Width of image in pixels.

height Height of image in pixels.

stride Stride of each row of the data space in bytes.

data Pointer to the image data.

RETURN VALUES The function returns a pointer to the mllib_image data structure.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageCreate(3MLIB), mllib_ImageCreateSubimage(3MLIB), mllib_ImageDelete(3MLIB), mllib_ImageSetPaddings(3MLIB), attributes(5)

mllib_ImageCreateSubimage(3MLIB)

NAME	mllib_ImageCreateSubimage – subimage creation										
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_image *mllib_ImageCreateSubimage(mllib_image *img, mllib_s32 x, mllib_s32 y, mllib_s32 w, mllib_s32 h);</pre>										
DESCRIPTION	The mllib_ImageCreateSubimage() function creates a mediaLib image data structure for a subimage based on a source image. Note that the memory space of the source image data is used for the subimage data. The data type of the image can be MLIB_BIT, MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, MLIB_INT, MLIB_FLOAT, or MLIB_DOUBLE.										
PARAMETERS	The function takes the following arguments: <table><tr><td><i>img</i></td><td>Pointer to source image.</td></tr><tr><td><i>x</i></td><td>X coordinate of subimage origin in the source.</td></tr><tr><td><i>y</i></td><td>Y coordinate of subimage origin in the source.</td></tr><tr><td><i>w</i></td><td>Width of the subimage in pixels.</td></tr><tr><td><i>h</i></td><td>Height of the subimage in pixels.</td></tr></table>	<i>img</i>	Pointer to source image.	<i>x</i>	X coordinate of subimage origin in the source.	<i>y</i>	Y coordinate of subimage origin in the source.	<i>w</i>	Width of the subimage in pixels.	<i>h</i>	Height of the subimage in pixels.
<i>img</i>	Pointer to source image.										
<i>x</i>	X coordinate of subimage origin in the source.										
<i>y</i>	Y coordinate of subimage origin in the source.										
<i>w</i>	Width of the subimage in pixels.										
<i>h</i>	Height of the subimage in pixels.										
RETURN VALUES	The function returns a pointer to the mllib_image data structure.										
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe				
ATTRIBUTE TYPE	ATTRIBUTE VALUE										
Interface Stability	Evolving										
MT-Level	MT-Safe										
SEE ALSO	mllib_ImageCreate(3MLIB), mllib_ImageCreateStruct(3MLIB), mllib_ImageDelete(3MLIB), mllib_ImageSetPaddings(3MLIB), attributes(5)										

NAME mllib_ImageCrossCorrel – cross correlation

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageCrossCorrel(mllib_d64 *correl, const
    mllib_image *img1, const mllib_image *img2);
```

DESCRIPTION The mllib_ImageCrossCorrel() function computes the cross-correlation between a pair of images.

It uses the following equation:

$$\text{correl}[i] = \frac{1}{w \cdot h} * \sum_{x=0}^{w-1} \sum_{y=0}^{h-1} (\text{img1}[x][y][i] * \text{img2}[x][y][i])$$

where w and h are the width and height of the images, respectively.

PARAMETERS The function takes the following arguments:

correl Pointer to cross correlation array on a channel basis. The array must be the size of channels in the images. *correl[i]* contains the cross-correlation of channel i.

img1 Pointer to first image.

img2 Pointer to second image.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageAutoCorrel(3MLIB), mllib_ImageAutoCorrel_Fp(3MLIB), mllib_ImageCrossCorrel_Fp(3MLIB), attributes(5)

mllib_ImageCrossCorrel_Fp(3MLIB)

NAME | mllib_ImageCrossCorrel_Fp – cross correlation

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageCrossCorrel_Fp(mllib_d64 *correl, const
    mllib_image *img1, const mllib_image *img2);
```

DESCRIPTION | The `mllib_ImageCrossCorrel_Fp()` function computes the cross-correlation between a pair of floating-point images.

It uses the following equation:

$$\text{correl}[i] = \frac{1}{w*h} * \sum_{x=0}^{w-1} \sum_{y=0}^{h-1} (\text{img1}[x][y][i] * \text{img2}[x][y][i])$$

where `w` and `h` are the width and height of the images, respectively.

PARAMETERS | The function takes the following arguments:

correl | Pointer to cross correlation array on a channel basis. The array must be the size of channels in the images. `correl[i]` contains the cross-correlation of channel `i`.

img1 | Pointer to first image.

img2 | Pointer to second image.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageAutoCorrel(3MLIB)`, `mllib_ImageAutoCorrel_Fp(3MLIB)`, `mllib_ImageCrossCorrel(3MLIB)`, `attributes(5)`

mllib_ImageDataTypeConvert(3MLIB)

NAME mllib_ImageDataTypeConvert – data type conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageDataTypeConvert(mllib_image *dst, const
    mllib_image *src);
```

DESCRIPTION The `mllib_ImageDataTypeConvert()` function converts between data types `MLIB_BIT`, `MLIB_BYTE`, `MLIB_SHORT`, `MLIB_USHORT`, `MLIB_INT`, `MLIB_FLOAT`, and `MLIB_DOUBLE`.

The input and output data images must have the same width, height, and number of channels. Conversion to a smaller pixel format clamps the source value to the dynamic range of the destination pixel.

See the following table for available variations of the data type conversion function.

Source Type	Dest. Type	Action
MLIB_BYTE	MLIB_BIT	(x > 0) ? 1 : 0
MLIB_SHORT	MLIB_BIT	(x > 0) ? 1 : 0
MLIB_USHORT	MLIB_BIT	(x > 0) ? 1 : 0
MLIB_INT	MLIB_BIT	(x > 0) ? 1 : 0
MLIB_FLOAT	MLIB_BIT	(x > 0) ? 1 : 0
MLIB_DOUBLE	MLIB_BIT	(x > 0) ? 1 : 0
MLIB_BIT	MLIB_BYTE	(x == 1) ? 1 : 0
MLIB_SHORT	MLIB_BYTE	(mllib_u8)clamp(x, 0, 255)
MLIB_USHORT	MLIB_BYTE	(mllib_u8)clamp(x, 0, 255)
MLIB_INT	MLIB_BYTE	(mllib_u8)clamp(x, 0, 255)
MLIB_FLOAT	MLIB_BYTE	(mllib_u8)clamp(x, 0, 255)
MLIB_DOUBLE	MLIB_BYTE	(mllib_u8)clamp(x, 0, 255)
MLIB_BIT	MLIB_SHORT	(x == 1) ? 1 : 0
MLIB_BYTE	MLIB_SHORT	(mllib_s16)x
MLIB_USHORT	MLIB_SHORT	(mllib_s16)clamp(x, -32768, 32767)
MLIB_INT	MLIB_SHORT	(mllib_s16)clamp(x, -32768, 32767)

mllib_ImageDataTypeConvert(3MLIB)

Source Type	Dest. Type	Action
MLIB_FLOAT	MLIB_SHORT	(mllib_s16)clamp(x, -32768, 32767)
MLIB_DOUBLE	MLIB_SHORT	(mllib_s16)clamp(x, -32768, 32767)
MLIB_BIT	MLIB_USHORT	(x == 1) ? 1 : 0
MLIB_BYTE	MLIB_USHORT	(mllib_u16)x
MLIB_SHORT	MLIB_USHORT	(mllib_u16)clamp(x, 0, 65535)
MLIB_INT	MLIB_USHORT	(mllib_u16)clamp(x, 0, 65535)
MLIB_FLOAT	MLIB_USHORT	(mllib_u16)clamp(x, 0, 65535)
MLIB_DOUBLE	MLIB_USHORT	(mllib_u16)clamp(x, 0, 65535)
MLIB_BIT	MLIB_INT	(x == 1) ? 1 : 0
MLIB_BYTE	MLIB_INT	(mllib_s32)x
MLIB_SHORT	MLIB_INT	(mllib_s32)x
MLIB_USHORT	MLIB_INT	(mllib_s32)x
MLIB_FLOAT	MLIB_INT	(mllib_s32)clamp(x, -2147483647-1, 2147483647)
MLIB_DOUBLE	MLIB_INT	(mllib_s32)clamp(x, -2147483647-1, 2147483647)
MLIB_BIT	MLIB_FLOAT	(x == 1) ? 1.0 : 0.0
MLIB_BYTE	MLIB_FLOAT	(mllib_f32)x
MLIB_SHORT	MLIB_FLOAT	(mllib_f32)x
MLIB_USHORT	MLIB_FLOAT	(mllib_f32)x
MLIB_INT	MLIB_FLOAT	(mllib_f32)x
MLIB_DOUBLE	MLIB_FLOAT	(mllib_f32)x
MLIB_BIT	MLIB_DOUBLE	(x == 1) ? 1.0 : 0.0
MLIB_BYTE	MLIB_DOUBLE	(mllib_d64)x
MLIB_SHORT	MLIB_DOUBLE	(mllib_d64)x
MLIB_USHORT	MLIB_DOUBLE	(mllib_d64)x
MLIB_INT	MLIB_DOUBLE	(mllib_d64)x
MLIB_FLOAT	MLIB_DOUBLE	(mllib_d64)x

mllib_ImageDataTypeConvert(3MLIB)

The actions are defined in C-style pseudo-code. All type casts follow the rules of standard C. `clamp()` can be defined as a macro: `#define clamp(x, low, high) (((x) < (low)) ? (low) : (((x) > (high)) ? (high) : (x)))`

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_ImageReformat\(3MLIB\)](#), [attributes\(5\)](#)

mllib_ImageDelete(3MLIB)

NAME | mllib_ImageDelete – image delete

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>
```



```
void mllib_ImageDelete(mllib_image *img);
```

DESCRIPTION | The `mllib_ImageDelete()` function deletes the mediaLib image data structure and frees the memory space of the image data only if it is allocated through `mllib_ImageCreate()`. The data type of the image can be `MLIB_BIT`, `MLIB_BYTE`, `MLIB_SHORT`, `MLIB_USHORT`, `MLIB_INT`, `MLIB_FLOAT`, or `MLIB_DOUBLE`.

PARAMETERS | The function takes the following arguments:
img Pointer to mediaLib image structure.

RETURN VALUES | None.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageCreate(3MLIB)`, `mllib_ImageCreateStruct(3MLIB)`,
`mllib_ImageCreateSubimage(3MLIB)`, `attributes(5)`

NAME	mllib_ImageDilate4 – four neighbor dilate						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageDilate4(mllib_image *dst, const mllib_image *src);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageDilate4()</code> function performs a dilation operation on an image by using each pixel's four orthogonal neighbors. The source and destination images must be single-channel images. The data type can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p> <p>For 4-neighbor binary images, it uses the following equation:</p> $\text{dst}[x][y][0] = \text{OR}\{ \text{src}[x][y][0], \text{src}[x-1][y][0], \text{src}[x+1][y][0], \text{src}[x][y-1][0], \text{src}[x][y+1][0] \}$ <p>For 4-neighbor grayscale images, it uses the following equation:</p> $\text{dst}[x][y][0] = \text{MAX}\{ \text{src}[x][y][0], \text{src}[x-1][y][0], \text{src}[x+1][y][0], \text{src}[x][y-1][0], \text{src}[x][y+1][0] \}$ <p>where $x = 1, \dots, w-2$; $y = 1, \dots, h-2$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageDilate4_Fp(3MLIB)</code> , <code>mllib_ImageDilate8(3MLIB)</code> , <code>mllib_ImageDilate8_Fp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageDilate4_Fp(3MLIB)

NAME	mllib_ImageDilate4_Fp – four neighbor dilate						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageDilate4_Fp(mllib_image *dst, const mllib_image *src);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageDilate4_Fp()</code> function performs a floating-point dilation operation on an image by using each pixel's four orthogonal neighbors. The source and destination images must be single-channel images. The data type can be <code>MLIB_FLOAT</code> or <code>MLIB_DOUBLE</code>.</p> <p>For 4-neighbor grayscale images, it uses the following equation:</p> $\text{dst}[x][y][0] = \text{MAX}\{ \text{src}[x][y][0], \text{src}[x-1][y][0], \text{src}[x+1][y][0], \text{src}[x][y-1][0], \text{src}[x][y+1][0] \}$ <p>where $x = 1, \dots, w-2$; $y = 1, \dots, h-2$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageDilate4(3MLIB)</code> , <code>mllib_ImageDilate8(3MLIB)</code> , <code>mllib_ImageDilate8_Fp(3MLIB)</code> , <code>attributes(5)</code>						

NAME	mllib_ImageDilate8 – eight neighbor dilate						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageDilate8(mllib_image *dst, const mllib_image *src);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageDilate8()</code> function performs a dilation operation on an image by using all eight of each pixel's neighbors. The source and destination images must be single-channel images. The data type can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p> <p>For 8-neighbor binary images, it uses the following equation:</p> $dst[x][y][0] = OR\{ src[p][q][0], x-1 \leq p \leq x+1; y-1 \leq q \leq y+1 \}$ <p>For 8-neighbor grayscale images, it uses the following equation:</p> $dst[x][y][0] = MAX\{ src[p][q][0], x-1 \leq p \leq x+1; y-1 \leq q \leq y+1 \}$ <p>where $x = 1, \dots, w-2$; $y = 1, \dots, h-2$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageDilate4(3MLIB)</code> , <code>mllib_ImageDilate4_Fp(3MLIB)</code> , <code>mllib_ImageDilate8_Fp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageDilate8_Fp(3MLIB)

NAME | mllib_ImageDilate8_Fp – eight neighbor dilate

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageDilate8_Fp(mllib_image *dst, const
    mllib_image *src);
```

DESCRIPTION | The `mllib_ImageDilate8_Fp()` function performs a floating-point dilation operation on an image by using all eight of each pixel's neighbors. The source and destination images must be single-channel images. The data type can be `MLIB_FLOAT` or `MLIB_DOUBLE`.

For 8-neighbor grayscale images, it uses the following equation:

$$dst[x][y][0] = \text{MAX}\{ src[p][q][0], x-1 \leq p \leq x+1; y-1 \leq q \leq y+1 \}$$

where $x = 1, \dots, w-2$; $y = 1, \dots, h-2$.

PARAMETERS | The function takes the following arguments:

dst | Pointer to destination image.

src | Pointer to source image.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageDilate4(3MLIB)`, `mllib_ImageDilate4_Fp(3MLIB)`, `mllib_ImageDilate8(3MLIB)`, `attributes(5)`

NAME	mllib_ImageDiv1_Fp_Inp – division, in place
SYNOPSIS	<pre>cc [flag...] file... -lmlib [library...] #include <mlib.h> mllib_status mllib_ImageDiv1_Fp_Inp(mllib_image *src1dst, const mllib_image *src2);</pre>
DESCRIPTION	<p>The <code>mllib_ImageDiv1_Fp_Inp()</code> function divides the second floating-point source image into the first floating-point source image on a pixel-by-pixel basis, in place.</p> <p>It uses the following equation:</p> $\text{src1dst}[x][y][i] = \text{src1dst}[x][y][i] / \text{src2}[x][y][i]$ <p>where the operation follows the IEEE-754 standard.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>src1dst</i> Pointer to first source and destination image.</p> <p><i>src2</i> Pointer to second source image.</p>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageDiv_Fp(3MLIB)`, `mllib_ImageDiv2_Fp_Inp(3MLIB)`, `attributes(5)`

mllib_ImageDiv2_Fp_Inp(3MLIB)

NAME | mllib_ImageDiv2_Fp_Inp – division, in place

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageDiv2_Fp_Inp(mllib_image *src2dst, const
    mllib_image *src1);
```

DESCRIPTION | The `mllib_ImageDiv2_Fp_Inp()` function divides the second floating-point source image into the first floating-point source image on a pixel-by-pixel basis, in place.

It uses the following equation:

$$\text{src2dst}[x][y][i] = \text{src1}[x][y][i] / \text{src2dst}[x][y][i]$$

where the operation follows the IEEE-754 standard.

PARAMETERS | The function takes the following arguments:

src2dst | Pointer to second source and destination image.

src1 | Pointer to first source image.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageDiv_Fp(3MLIB)`, `mllib_ImageDiv1_Fp_Inp(3MLIB)`, `attributes(5)`

NAME	mllib_ImageDivAlpha – alpha channel division
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageDivAlpha(mllib_image *dst, const mllib_image *src, mllib_s32 cmask);</pre>
DESCRIPTION	<p>The <code>mllib_ImageDivAlpha()</code> function divides color channels by the alpha channel on a pixel-by-pixel basis.</p> <p>For the <code>MLIB_BYTE</code> image, it uses the following equation:</p> $\text{dst}[x][y][c] = \text{src}[x][y][c] / (\text{src}[x][y][a] * 2^{**(-8)})$ <p>For the <code>MLIB_SHORT</code> image, it uses the following equation:</p> $\text{dst}[x][y][c] = \text{src}[x][y][c] / (\text{src}[x][y][a] * 2^{**(-15)})$ <p>For the <code>MLIB_USHORT</code> image, it uses the following equation:</p> $\text{dst}[x][y][c] = \text{src}[x][y][c] / (\text{src}[x][y][a] * 2^{**(-16)})$ <p>For the <code>MLIB_INT</code> image, it uses the following equation:</p> $\text{dst}[x][y][c] = \text{src}[x][y][c] / (\text{src}[x][y][a] * 2^{**(-31)})$ <p>where <code>c</code> and <code>a</code> are the indices for the color channels and the alpha channel, respectively, so <code>c != a</code>.</p> <p>In the case of <code>src[x][y][a] = 0</code>,</p> <pre>dst[x][y][c] = 0 if src[x][y][c] = 0 dst[x][y][c] = DATA_TYPE_MAX if src[x][y][c] > 0 dst[x][y][c] = DATA_TYPE_MIN if src[x][y][c] < 0</pre> <p>where <code>DATA_TYPE</code> is <code>MLIB_U8</code>, <code>MLIB_S16</code>, <code>MLIB_U16</code>, or <code>MLIB_S32</code> for an image of type <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>, respectively.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>cmask</i> Channel mask to indicate the alpha channel. Each bit of the mask represents a channel in the image. The channel corresponding to the 1 bit of <code>cmask</code> is the alpha channel.</p>
RETURN VALUES	<p>The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code>.</p>

mllib_ImageDivAlpha(3MLIB)

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageDivAlpha_Inp(3MLIB)`, `mllib_ImageDivAlpha_Fp(3MLIB)`,
`mllib_ImageDivAlpha_Fp_Inp(3MLIB)`, `attributes(5)`

NAME mllib_ImageDivAlpha_Fp – alpha channel division

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageDivAlpha_Fp(mllib_image *dst, const
    mllib_image *src, mllib_s32 cmask);
```

DESCRIPTION The mllib_ImageDivAlpha_Fp() function divides floating-point color channels by the alpha channel on a pixel-by-pixel basis.

It uses the following equation:

$$dst[x][y][c] = src[x][y][c] / src[x][y][a]$$

where c and a are the indices for the color channels and the alpha channel, respectively, so $c \neq a$.

The operation follows the IEEE-754 standard.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

cmask Channel mask to indicate the alpha channel. Each bit of the mask represents a channel in the image. The channel corresponding to the 1 bit of *cmask* is the alpha channel.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageDivAlpha(3MLIB), mllib_ImageDivAlpha_Fp_Inp(3MLIB), mllib_ImageDivAlpha_Inp(3MLIB), attributes(5)

mllib_ImageDivAlpha_Fp_Inp(3MLIB)

NAME	mllib_ImageDivAlpha_Fp_Inp – alpha channel division, in place						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageDivAlpha_Fp_Inp(mllib_image *srcdst, mllib_s32 cmask);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageDivAlpha_Fp_Inp()</code> function divides floating-point color channels by the alpha channel on a pixel-by-pixel basis, in place.</p> <p>It uses the following equation:</p> $\text{srcdst}[x][y][c] = \text{srcdst}[x][y][c] / \text{srcdst}[x][y][a]$ <p>where <code>c</code> and <code>a</code> are the indices for the color channels and the alpha channel, respectively, so <code>c != a</code>.</p> <p>The operation follows the IEEE-754 standard.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>srcdst</i></td><td>Pointer to source and destination image.</td></tr><tr><td><i>cmask</i></td><td>Channel mask to indicate the alpha channel. Each bit of the mask represents a channel in the image. The channel corresponding to the 1 bit of <i>cmask</i> is the alpha channel.</td></tr></table>	<i>srcdst</i>	Pointer to source and destination image.	<i>cmask</i>	Channel mask to indicate the alpha channel. Each bit of the mask represents a channel in the image. The channel corresponding to the 1 bit of <i>cmask</i> is the alpha channel.		
<i>srcdst</i>	Pointer to source and destination image.						
<i>cmask</i>	Channel mask to indicate the alpha channel. Each bit of the mask represents a channel in the image. The channel corresponding to the 1 bit of <i>cmask</i> is the alpha channel.						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageDivAlpha(3MLIB)</code> , <code>mllib_ImageDivAlpha_Fp(3MLIB)</code> , <code>mllib_ImageDivAlpha_Inp(3MLIB)</code> , <code>attributes(5)</code>						

NAME	mllib_ImageDivAlpha_Inp – alpha channel division, in place
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageDivAlpha_Inp(mllib_image *srcdst, mllib_s32 cmask);</pre>
DESCRIPTION	<p>The <code>mllib_ImageDivAlpha_Inp()</code> function divides color channels by the alpha channel on a pixel-by-pixel basis, in place.</p> <p>For the <code>MLIB_BYTE</code> image, it uses the following equation:</p> $\text{srcdst}[x][y][c] = \text{srcdst}[x][y][c] / (\text{srcdst}[x][y][a] * 2^{**(-8)})$ <p>For the <code>MLIB_SHORT</code> image, it uses the following equation:</p> $\text{srcdst}[x][y][c] = \text{srcdst}[x][y][c] / (\text{srcdst}[x][y][a] * 2^{**(-15)})$ <p>For the <code>MLIB_USHORT</code> image, it uses the following equation:</p> $\text{srcdst}[x][y][c] = \text{srcdst}[x][y][c] / (\text{srcdst}[x][y][a] * 2^{**(-16)})$ <p>For the <code>MLIB_INT</code> image, it uses the following equation:</p> $\text{srcdst}[x][y][c] = \text{srcdst}[x][y][c] / (\text{srcdst}[x][y][a] * 2^{**(-31)})$ <p>where <code>c</code> and <code>a</code> are the indices for the color channels and the alpha channel, respectively, so <code>c != a</code>.</p> <p>In the case of <code>srcdst[x][y][a] = 0</code>,</p> <pre>srcdst[x][y][c] = 0 if srcdst[x][y][c] = 0 srcdst[x][y][c] = DATA_TYPE_MAX if srcdst[x][y][c] > 0 srcdst[x][y][c] = DATA_TYPE_MIN if srcdst[x][y][c] < 0</pre> <p>where <code>DATA_TYPE</code> is <code>MLIB_U8</code>, <code>MLIB_S16</code>, <code>MLIB_U16</code>, or <code>MLIB_S32</code> for an image of type <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>, respectively.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>srcdst</i> Pointer to source and destination image.</p> <p><i>cmask</i> Channel mask to indicate the alpha channel. Each bit of the mask represents a channel in the image. The channel corresponding to the 1 bit of <i>cmask</i> is the alpha channel.</p>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:

mllib_ImageDivAlpha_Inp(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_ImageDivAlpha\(3MLIB\)](#), [mllib_ImageDivAlpha_Fp\(3MLIB\)](#),
[mllib_ImageDivAlpha_Fp_Inp\(3MLIB\)](#), [attributes\(5\)](#)

NAME mllib_ImageDivConstShift – division by a constant, with shifting

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageDivConstShift(mllib_image *dst, const
    mllib_image *src, const mllib_s32 *c, mllib_s32 shift);
```

DESCRIPTION The `mllib_ImageDivConstShift()` function divides each pixel in an image by a constant value on a pixel-by-pixel basis. It scales the result by a left shift and writes the result to the destination image on a pixel-by-pixel basis.

It uses the following equation:

$$dst[x][y][i] = src[x][y][i] / c[i] * 2^{**shift}$$

In the case of $c[i] = 0$,

$$dst[x][y][i] = 0 \quad \text{if } src[x][y][i] = 0$$

$$dst[x][y][i] = DATA_TYPE_MAX \quad \text{if } src[x][y][i] > 0$$

$$dst[x][y][i] = DATA_TYPE_MIN \quad \text{if } src[x][y][i] < 0$$

where `DATA_TYPE` is `MLIB_U8`, `MLIB_S16`, `MLIB_U16`, or `MLIB_S32` for an image of type `MLIB_BYTE`, `MLIB_SHORT`, `MLIB_USHORT`, or `MLIB_INT`, respectively.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

c Constant by which each pixel is divided. `c[i]` contains the constant for channel *i*.

shift Left shifting factor. $0 \leq shift \leq 31$.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageDivConstShift_Inp(3MLIB)`, `attributes(5)`

mllib_ImageDivConstShift_Inp(3MLIB)

NAME	mllib_ImageDivConstShift_Inp – division by a constant, with shifting						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageDivConstShift_Inp(mllib_image *srcdst, const mllib_s32 *c, mllib_s32 shift);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageDivConstShift_Inp()</code> function divides each pixel in an image by a constant value on a pixel-by-pixel basis, in place. It scales the result by a left shift and writes the result to the image on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $\text{srcdst}[x][y][i] = \text{srcdst}[x][y][i] / c[i] * 2^{**\text{shift}}$ <p>In the case of $c[i] = 0$,</p> $\begin{aligned} \text{srcdst}[x][y][i] &= 0 && \text{if } \text{srcdst}[x][y][i] = 0 \\ \text{srcdst}[x][y][i] &= \text{DATA_TYPE_MAX} && \text{if } \text{srcdst}[x][y][i] > 0 \\ \text{srcdst}[x][y][i] &= \text{DATA_TYPE_MIN} && \text{if } \text{srcdst}[x][y][i] < 0 \end{aligned}$ <p>where <code>DATA_TYPE</code> is <code>MLIB_U8</code>, <code>MLIB_S16</code>, <code>MLIB_U16</code>, or <code>MLIB_S32</code> for an image of type <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>, respectively.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>srcdst</i></td><td>Pointer to source and destination image.</td></tr><tr><td><i>c</i></td><td>Constant by which each pixel is divided. <code>c[i]</code> contains the constant for channel <i>i</i>.</td></tr><tr><td><i>shift</i></td><td>Left shifting factor. $0 \leq \text{shift} \leq 31$.</td></tr></table>	<i>srcdst</i>	Pointer to source and destination image.	<i>c</i>	Constant by which each pixel is divided. <code>c[i]</code> contains the constant for channel <i>i</i> .	<i>shift</i>	Left shifting factor. $0 \leq \text{shift} \leq 31$.
<i>srcdst</i>	Pointer to source and destination image.						
<i>c</i>	Constant by which each pixel is divided. <code>c[i]</code> contains the constant for channel <i>i</i> .						
<i>shift</i>	Left shifting factor. $0 \leq \text{shift} \leq 31$.						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageDivConstShift(3MLIB)</code> , <code>attributes(5)</code>						

NAME	mlib_ImageDiv_Fp – division						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmlib [<i>library...</i>] #include <mlib.h> mlib_status mlib_ImageDiv_Fp(mlib_image *<i>dst</i>, const mlib_image *<i>src1</i>, const mlib_image *<i>src2</i>);</pre>						
DESCRIPTION	<p>The <code>mlib_ImageDiv_Fp()</code> function divides the second floating-point source image into the first floating-point source image on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $dst[x][y][i] = src1[x][y][i] / src2[x][y][i]$ <p>where the operation follows the IEEE-754 standard.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src1</i> Pointer to first source image.</p> <p><i>src2</i> Pointer to second source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mlib_ImageDiv1_Fp_Inp(3MLIB)</code> , <code>mlib_ImageDiv2_Fp_Inp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageDivShift1_Inp(3MLIB)

NAME	mllib_ImageDivShift1_Inp – division with shifting, in place						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageDivShift1_Inp(mllib_image *src1dst, const mllib_image *src2, mllib_s32 shift);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageDivShift1_Inp()</code> function divides the second source image into the first source image on a pixel-by-pixel basis. It scales the result by a left shift and writes the result to the destination image on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $\text{src1dst}[x][y][i] = \text{src1dst}[x][y][i] / \text{src2}[x][y][i] * 2^{**\text{shift}}$ <p>In the case of $\text{src2}[x][y][i] = 0$,</p> $\begin{aligned} \text{src1dst}[x][y][i] &= 0 && \text{if } \text{src1dst}[x][y][i] = 0 \\ \text{src1dst}[x][y][i] &= \text{DATA_TYPE_MAX} && \text{if } \text{src1dst}[x][y][i] > 0 \\ \text{src1dst}[x][y][i] &= \text{DATA_TYPE_MIN} && \text{if } \text{src1dst}[x][y][i] < 0 \end{aligned}$ <p>where <code>DATA_TYPE</code> is <code>MLIB_U8</code>, <code>MLIB_S16</code>, <code>MLIB_U16</code>, or <code>MLIB_S32</code> for an image of type <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>, respectively.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>src1dst</i></td><td>Pointer to first source and destination image.</td></tr><tr><td><i>src2</i></td><td>Pointer to second source image.</td></tr><tr><td><i>shift</i></td><td>Left shifting factor. $0 \leq \text{shift} \leq 31$.</td></tr></table>	<i>src1dst</i>	Pointer to first source and destination image.	<i>src2</i>	Pointer to second source image.	<i>shift</i>	Left shifting factor. $0 \leq \text{shift} \leq 31$.
<i>src1dst</i>	Pointer to first source and destination image.						
<i>src2</i>	Pointer to second source image.						
<i>shift</i>	Left shifting factor. $0 \leq \text{shift} \leq 31$.						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageDivShift(3MLIB)</code> , <code>mllib_ImageDivShift2_Inp(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageDivShift2_Inp – division with shifting, in place

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageDivShift2_Inp(mllib_image *src2dst, const
    mllib_image *src1, mllib_s32 shift);
```

DESCRIPTION The mllib_ImageDivShift2_Inp() function divides the second source image into the first source image on a pixel-by-pixel basis. It scales the result by a left shift and writes the result to the destination image on a pixel-by-pixel basis.

It uses the following equation:

$$src2dst[x][y][i] = src1[x][y][i] / src2dst[x][y][i] * 2^{**shift}$$

In the case of $src2dst[x][y][i] = 0$,

$$src2dst[x][y][i] = 0 \quad \text{if } src1[x][y][i] = 0$$

$$src2dst[x][y][i] = DATA_TYPE_MAX \quad \text{if } src1[x][y][i] > 0$$

$$src2dst[x][y][i] = DATA_TYPE_MIN \quad \text{if } src1[x][y][i] < 0$$

where DATA_TYPE is MLIB_U8, MLIB_S16, MLIB_U16, or MLIB_S32 for an image of type MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, or MLIB_INT, respectively.

PARAMETERS The function takes the following arguments:

src2dst Pointer to second source and destination image.

src1 Pointer to first source image.

shift Left shifting factor. $0 \leq shift \leq 31$.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageDivShift(3MLIB), mllib_ImageDivShift1_Inp(3MLIB), attributes(5)

mllib_ImageDivShift(3MLIB)

NAME	mllib_ImageDivShift – division with shifting								
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageDivShift(mllib_image *dst, const mllib_image *src1, const mllib_image *src2, mllib_s32 shift);</pre>								
DESCRIPTION	<p>The <code>mllib_ImageDivShift()</code> function divides the second source image into the first source image on a pixel-by-pixel basis. It scales the result by a left shift and writes the result to the destination image on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $\text{dst}[x][y][i] = \text{src1}[x][y][i] / \text{src2}[x][y][i] * 2^{**\text{shift}}$ <p>In the case of $\text{src2}[x][y][i] = 0$,</p> $\begin{aligned} \text{dst}[x][y][i] &= 0 && \text{if } \text{src1}[x][y][i] = 0 \\ \text{dst}[x][y][i] &= \text{DATA_TYPE_MAX} && \text{if } \text{src1}[x][y][i] > 0 \\ \text{dst}[x][y][i] &= \text{DATA_TYPE_MIN} && \text{if } \text{src1}[x][y][i] < 0 \end{aligned}$ <p>where <code>DATA_TYPE</code> is <code>MLIB_U8</code>, <code>MLIB_S16</code>, <code>MLIB_U16</code>, or <code>MLIB_S32</code> for an image of type <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>, respectively.</p>								
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>dst</i></td><td>Pointer to destination image.</td></tr><tr><td><i>src1</i></td><td>Pointer to first source image.</td></tr><tr><td><i>src2</i></td><td>Pointer to second source image.</td></tr><tr><td><i>shift</i></td><td>Left shifting factor. $0 \leq \text{shift} \leq 31$.</td></tr></table>	<i>dst</i>	Pointer to destination image.	<i>src1</i>	Pointer to first source image.	<i>src2</i>	Pointer to second source image.	<i>shift</i>	Left shifting factor. $0 \leq \text{shift} \leq 31$.
<i>dst</i>	Pointer to destination image.								
<i>src1</i>	Pointer to first source image.								
<i>src2</i>	Pointer to second source image.								
<i>shift</i>	Left shifting factor. $0 \leq \text{shift} \leq 31$.								
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .								
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:								
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe		
ATTRIBUTE TYPE	ATTRIBUTE VALUE								
Interface Stability	Evolving								
MT-Level	MT-Safe								
SEE ALSO	<code>mllib_ImageDivShift1_Inp(3MLIB)</code> , <code>mllib_ImageDivShift2_Inp(3MLIB)</code> , <code>attributes(5)</code>								

NAME	mllib_ImageErode4 – four neighbor erode						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageErode4(mllib_image *dst, const mllib_image *src);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageErode4()</code> function performs an erode operation on an image by using each pixel's four orthogonal neighbors. The source and destination images must be single-channel images. The data type can be <code>MLIB_BIT</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p> <p>For 4-neighbor binary images, it uses the following equation:</p> $\text{dst}[x][y][0] = \text{AND}\{ \text{src}[x][y][0], \text{src}[x-1][y][0], \text{src}[x+1][y][0], \text{src}[x][y-1][0], \text{src}[x][y+1][0] \}$ <p>For 4-neighbor grayscale images, it uses the following equation:</p> $\text{dst}[x][y][0] = \text{MIN}\{ \text{src}[x][y][0], \text{src}[x-1][y][0], \text{src}[x+1][y][0], \text{src}[x][y-1][0], \text{src}[x][y+1][0] \}$ <p>where $x = 1, \dots, w-2$; $y = 1, \dots, h-2$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageErode4_Fp(3MLIB)</code> , <code>mllib_ImageErode8(3MLIB)</code> , <code>mllib_ImageErode8_Fp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageErode4_Fp(3MLIB)

NAME	mllib_ImageErode4_Fp – four neighbor erode						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageErode4_Fp(mllib_image *dst, const mllib_image *src);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageErode4_Fp()</code> function performs an erode operation on an image by using each pixel's four orthogonal neighbors. The source and destination images must be single-channel images. The data type of the images can be <code>MLIB_FLOAT</code> or <code>MLIB_DOUBLE</code>.</p> <p>For 4-neighbor grayscale images, it uses the following equation:</p> $\text{dst}[x][y][0] = \text{MIN}\{ \text{src}[x][y][0], \text{src}[x-1][y][0], \text{src}[x+1][y][0], \text{src}[x][y-1][0], \text{src}[x][y+1][0] \}$ <p>where $x = 1, \dots, w-2$; $y = 1, \dots, h-2$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageErode4(3MLIB)</code> , <code>mllib_ImageErode8(3MLIB)</code> , <code>mllib_ImageErode8_Fp(3MLIB)</code> , <code>attributes(5)</code>						

NAME	mllib_ImageErode8 – eight neighbor erode						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageErode8(mllib_image *dst, const mllib_image *src);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageErode8()</code> function performs an erode operation on an image by using all eight of each pixel's neighbors. The source and destination images must be single-channel images. The data type can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p> <p>For 8-neighbor binary images, it uses the following equation:</p> $dst[x][y][0] = \text{AND}\{ src[p][q][0], x-1 \leq p \leq x+1; y-1 \leq q \leq y+1 \}$ <p>For 8-neighbor grayscale images, it uses the following equation:</p> $dst[x][y][0] = \text{MIN}\{ src[p][q][0], x-1 \leq p \leq x+1; y-1 \leq q \leq y+1 \}$ <p>where $x = 1, \dots, w-2$; $y = 1, \dots, h-2$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageErode4(3MLIB)</code> , <code>mllib_ImageErode4_Fp(3MLIB)</code> , <code>mllib_ImageErode8_Fp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageErode8_Fp(3MLIB)

NAME | mllib_ImageErode8_Fp – eight neighbor erode

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageErode8_Fp(mllib_image *dst, const mllib_image
    *src);
```

DESCRIPTION | The `mllib_ImageErode8_Fp()` function performs an erode operation on an image by using all eight of each pixel's neighbors. The source and destination images must be single-channel images. The data type of the images can be `MLIB_FLOAT` or `MLIB_DOUBLE`.

For 8-neighbor grayscale images, it uses the following equation:

$$dst[x][y][0] = \text{MIN}\{ src[p][q][0], x-1 \leq p \leq x+1; y-1 \leq q \leq y+1 \}$$

where $x = 1, \dots, w-2$; $y = 1, \dots, h-2$.

PARAMETERS | The function takes the following arguments:

dst | Pointer to destination image.

src | Pointer to source image.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageErode4(3MLIB)`, `mllib_ImageErode4_Fp(3MLIB)`, `mllib_ImageErode8(3MLIB)`, `attributes(5)`

NAME mllib_ImageExp – computes the exponent of the image pixels

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageExp(mllib_image *dst, const mllib_image *src);
```

DESCRIPTION The mllib_ImageExp() function computes the exponent of the image pixels. It uses the following equation:
 $dst[x][y][i] = e^{**src[x][y][i]}$

PARAMETERS The function takes the following arguments:
dst Pointer to destination image.
src Pointer to source image.

RETURN VALUES The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_ImageExp_Fp\(3MLIB\)](#), [mllib_ImageExp_Fp_Inp\(3MLIB\)](#), [mllib_ImageExp_Inp\(3MLIB\)](#), [attributes\(5\)](#)

mllib_ImageExp_Fp(3MLIB)

NAME | mllib_ImageExp_Fp – computes the exponent of the image pixels

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
mllib_status mllib_ImageExp_Fp(mllib_image *dst, const mllib_image  
*src);
```

DESCRIPTION | The mllib_ImageExp_Fp() function computes the exponent of the floating-point image pixels.

It uses the following equation:
$$dst[x][y][i] = e**src[x][y][i]$$

PARAMETERS | The function takes the following arguments:

dst | Pointer to destination image.

src | Pointer to source image.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_ImageExp\(3MLIB\)](#), [mllib_ImageExp_Fp_Inp\(3MLIB\)](#), [mllib_ImageExp_Inp\(3MLIB\)](#), [attributes\(5\)](#)

NAME mllib_ImageExp_Fp_Inp – computes the exponent of the image pixels

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageExp_Fp_Inp(mllib_image *srcdst);
```

DESCRIPTION The mllib_ImageExp_Fp_Inp() function computes the exponent of the floating-point image pixels.

It uses the following equation:

$$srcdst[x][y][i] = e**srcdst[x][y][i]$$

PARAMETERS The function takes the following arguments:

srcdst Pointer to source and destination image.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_ImageExp\(3MLIB\)](#), [mllib_ImageExp_Fp\(3MLIB\)](#), [mllib_ImageExp_Inp\(3MLIB\)](#), [attributes\(5\)](#)

mllib_ImageExp_Inp(3MLIB)

NAME	mllib_ImageExp_Inp – computes the exponent of the image pixels
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageExp_Inp(mllib_image *srcdst);</pre>
DESCRIPTION	<p>The mllib_ImageExp_Inp() function computes the exponent of the image pixels, in place.</p> <p>It uses the following equation:</p> $\text{srcdst}[x][y][i] = e^{**\text{srcdst}[x][y][i]}$
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>srcdst</i> Pointer to source and destination image.</p>
RETURN VALUES	The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_ImageExp\(3MLIB\)](#), [mllib_ImageExp_Fp\(3MLIB\)](#), [mllib_ImageExp_Fp_Inp\(3MLIB\)](#), [attributes\(5\)](#)

NAME mllib_ImageExtrema2, mllib_ImageExtrema2_Fp – image extrema

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageExtrema2(mllib_s32 *min, mllib_s32 *max,
    const mllib_image *img, mllib_s32 xStart, mllib_s32 yStart, mllib_s32
    xPeriod, mllib_s32 yPeriod);

mllib_status mllib_ImageExtrema2_Fp(mllib_d64 *min, mllib_d64 *max,
    const mllib_image *img, mllib_s32 xStart, mllib_s32 yStart, mllib_s32
    xPeriod, mllib_s32 yPeriod);
```

DESCRIPTION Each of the functions determines the extrema values for each channel in an image, possibly with subsampling.

It uses the following equation:

$$\min[i] = \text{MIN}\{ \text{img}[x][y][i] \}$$

$$\max[i] = \text{MAX}\{ \text{img}[x][y][i] \}$$

where

$$x = xStart + p * xPeriod; \quad 0 \leq p < (w - xStart) / xPeriod$$

$$y = yStart + q * yPeriod; \quad 0 \leq q < (h - yStart) / yPeriod$$

PARAMETERS Each of the functions takes the following arguments:

min Pointer to minimum vector, where length is the number of channels in the image. *min*[*i*] contains the minimum of channel *i*.

max Pointer to maximum vector, where length is the number of channels in the image. *max*[*i*] contains the maximum of channel *i*.

img Pointer to a source image.

xStart Initial X sample coordinate.

yStart Initial Y sample coordinate.

xPeriod X sample rate. *xPeriod* ≥ 1.

yPeriod Y sample rate. *yPeriod* ≥ 1.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

mllib_ImageExtrema2(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	MT-Safe

SEE ALSO `mllib_ImageExtremaLocations(3MLIB)`, `mllib_ImageMaximum(3MLIB)`,
`mllib_ImageMaximum_Fp(3MLIB)`, `mllib_ImageMinimum(3MLIB)`,
`mllib_ImageMinimum_Fp(3MLIB)`, `attributes(5)`

NAME	mllib_ImageExtremaLocations, mllib_ImageExtremaLocations_Fp – image extrema and their locations
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageExtremaLocations(mllib_s32 *min, mllib_s32 *max, const mllib_image *img, mllib_s32 xStart, mllib_s32 yStart, mllib_s32 xPeriod, mllib_s32 yPeriod, mllib_s32 saveLocations, mllib_s32 maxRuns, mllib_s32 *minCounts, mllib_s32 *maxCounts, mllib_s32 **minLocations, mllib_s32 **maxLocations, mllib_s32 len); mllib_status mllib_ImageExtremaLocations_Fp(mllib_d64 *min, mllib_d64 *max, const mllib_image *img, mllib_s32 xStart, mllib_s32 yStart, mllib_s32 xPeriod, mllib_s32 yPeriod, mllib_s32 saveLocations, mllib_s32 maxRuns, mllib_s32 *minCounts, mllib_s32 *maxCounts, mllib_s32 **minLocations, mllib_s32 **maxLocations, mllib_s32 len);</pre>
DESCRIPTION	<p>Each of the functions finds the image-wise minimum and maximum pixel values for each channel, and optionally, their locations.</p> <p>Each of the functions scans an image, finds the minimum and maximum pixel values for each channel, and finds the locations of those pixels with the minimum or maximum values.</p> <p>The user provides initial minimum/maximum values through the arguments <code>min</code> and <code>max</code>. This function will update them based on findings.</p> <p>The set of pixels scanned may furthermore be reduced by specifying <code>xPeriod</code> and <code>yPeriod</code> parameters that specify the sampling rate along each axis.</p> <p>The set of pixels to be scanned may be obtained from the following equation:</p> <pre>x = xStart + p*xPeriod; 0 ≤ p < (w - xStart)/xPeriod y = yStart + q*yPeriod; 0 ≤ q < (h - yStart)/yPeriod</pre> <p>The locations of the minimum/maximum, if asked, are recorded in a format of run-length coding. Each run-length code, or simply called a run, has a format of (<code>xStart</code>, <code>yStart</code>, <code>length</code>). Here <code>length</code> is defined on the low-resolution image (with downsampling factors of $1/xPeriod$, $1/yPeriod$) and does not cross rows. So the run-length code (<code>xStart</code>, <code>yStart</code>, <code>length</code>) means that the pixels at (<code>xStart</code>, <code>yStart</code>), (<code>xStart</code> + <code>xPeriod</code>, <code>yStart</code>), ..., (<code>xStart</code> + (<code>length</code> - 1) * <code>xPeriod</code>, <code>yStart</code>) of the original image have a value of the minimum/maximum.</p> <p>The buffers for <code>minLocations</code> and <code>maxLocations</code> are organized in the following format for each channel <code>i</code>:</p> <pre>minLocations[i][0] = xStart0; // the 1st run minLocations[i][1] = yStart0; minLocations[i][2] = length0; minLocations[i][3] = xStart1; // the 2nd run minLocations[i][4] = yStart1;</pre>

mllib_ImageExtremaLocations(3MLIB)

```
minLocations[i][5] = length1;
    ..... // more runs
minLocations[i][len-1] = ...;
```

It is the user's responsibility to allocate enough memory for the buffers for `minLocations` and `maxLocations`. This function may return `MLIB_OUTOFRANGE`, if any of the buffers is not big enough.

PARAMETERS

The function takes the following arguments:

<i>min</i>	Pointer to the minimum values.
<i>max</i>	Pointer to the maximum values.
<i>img</i>	Pointer to the input image.
<i>xStart</i>	Initial X sample coordinate.
<i>yStart</i>	Initial Y sample coordinate.
<i>xPeriod</i>	X sampling rate. $xPeriod \geq 1$.
<i>yPeriod</i>	Y sampling rate. $yPeriod \geq 1$.
<i>saveLocations</i>	If true (i.e., <code>saveLocations != 0</code>), find the extrema locations; otherwise only find the extrema.
<i>maxRuns</i>	Number of runs of the minimum/maximum the caller expects for each channel. $maxRuns \geq 1$. If it is <code>MLIB_S32_MAX</code> , all the minimum/maximum locations should be recorded.
<i>minCounts</i>	Pointer to the numbers of runs of the minimum recorded in <code>minLocations</code> .
<i>maxCounts</i>	Pointer to the numbers of runs of the maximum recorded in <code>maxLocations</code> .
<i>minLocations</i>	Pointer to the minimum locations in a format of run-length coding.
<i>maxLocations</i>	Pointer to the maximum locations in a format of run-length coding.
<i>len</i>	Length of the buffers for the minimum/maximum locations in each channel.

RETURN VALUES

The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

`mllib_ImageExtremaLocations(3MLIB)`

SEE ALSO `mllib_ImageExtrema2(3MLIB)`, `mllib_ImageMaximum(3MLIB)`,
`mllib_ImageMaximum_Fp(3MLIB)`, `mllib_ImageMinimum(3MLIB)`,
`mllib_ImageMinimum_Fp(3MLIB)`, `attributes(5)`

mllib_ImageFilteredSubsample(3MLIB)

NAME	mllib_ImageFilteredSubsample, mllib_ImageFilteredSubsample_Fp – antialias filters and subsamples an image																																																							
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageFilteredSubsample(mllib_image *dst, const mllib_image *src, mllib_s32 scaleX, mllib_s32 scaleY, mllib_s32 transX, mllib_s32 transY, const mllib_d64 *hKernel, const mllib_d64 *vKernel, mllib_s32 hSize, mllib_s32 vSize, mllib_s32 hParity, mllib_s32 vParity, mllib_edge edge); mllib_status mllib_ImageFilteredSubsample_Fp(mllib_image *dst, const mllib_image *src, mllib_s32 scaleX, mllib_s32 scaleY, mllib_s32 transX, mllib_s32 transY, const mllib_d64 *hKernel, const mllib_d64 *vKernel, mllib_s32 hSize, mllib_s32 vSize, mllib_s32 hParity, mllib_s32 vParity, mllib_edge edge);</pre>																																																							
DESCRIPTION	<p>Each of the functions antialias filters and subsamples an image.</p> <p>The effect of one of the functions on an image is equivalent to performing convolution (filter) followed by subsampling (zoom out).</p> <p>The functions are similar to the <code>mllib_ImageZoomTranslate()</code> and <code>mllib_ImageZoomTranslate_Fp()</code> functions. But they have different definitions on scale factors and translations, hence use different coordinate mapping equations. The <code>scaleX</code> and <code>scaleY</code> used by <code>mllib_ImageFilteredSubsample()</code> and <code>mllib_ImageFilteredSubsample_Fp()</code> are the reciprocals of the <code>zoomx</code> and <code>zoomy</code>, respectively, used by <code>mllib_ImageZoomTranslate()</code> and <code>mllib_ImageZoomTranslate_Fp()</code>.</p> <p>The functions use the following equations for coordinate mapping:</p> $xS = xD * scaleX + transX$ $yS = yD * scaleY + transY$ <p>where, a point (xD, yD) in the destination image is backward mapped to a point (xS, yS) in the source image. The arguments <code>transX</code> and <code>transY</code> are provided to support tiling.</p> <p>The subsample terms, i.e., the scale factors <code>scaleX</code> and <code>scaleY</code>, are restricted to positive integral values. Geometrically, one destination pixel maps to <code>scaleX</code> by <code>scaleY</code> source pixels. With odd scale factors, destination pixel centers map directly onto source pixel centers. With even scale factors, destination pixel centers map squarely between source pixel centers. Below are examples of even, odd, and combination cases.</p> <table><tr><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td></tr><tr><td></td><td>d</td><td></td><td>d</td><td></td><td>d</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>d</td><td>s</td><td>s</td></tr><tr><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td><td>s</td></tr><tr><td></td><td>d</td><td></td><td>d</td><td></td><td>d</td><td></td><td></td><td></td><td></td><td></td></tr></table>	s	s	s	s	s	s	s	s	s	s	s		d		d		d						s	s	s	s	s	s	s	s	d	s	s	s	s	s	s	s	s	s	s	s	s	s		d		d		d					
s	s	s	s	s	s	s	s	s	s	s																																														
	d		d		d																																																			
s	s	s	s	s	s	s	s	d	s	s																																														
s	s	s	s	s	s	s	s	s	s	s																																														
	d		d		d																																																			

mllib_ImageFilteredSubsample(3MLIB)

```

s s s s s s s      s s s s s s s
s s s s s s s      s d s s d s
  d      d      d
s s s s s s s      s s s s s s s

Even scaleX/Y factors      Odd scaleX/Y factors

s s s s s s s      s s s s s s s
  d      d
s s s s s s s      s d s s d s s d s

s s s s s s s      s s s s s s s
  d      d
s s s s s s s      s s s s s s s

s s s s s s s      s d s s d s s d s
  d      d
s s s s s s s      s s s s s s s

Odd/even scaleX/Y factors      Even/odd scaleX/Y factors

```

where

```

s = source pixel centers
d = destination pixel centers mapped to source

```

The applied filter is quadrant symmetric (typically antialias + resample). The filter is product-separable, quadrant symmetric, and is defined by half of its span. Parity is used to signify whether the symmetric kernel has a double center (even parity) or a single center value (odd parity). For example, if `hParity == 0` (even), the horizontal kernel is defined as:

```

hKernel[hSize-1], ..., hKernel[0], hKernel[0], ...,
hKernel[hSize-1]

```

Otherwise, if `hParity == 1` (odd), the horizontal kernel is defined as:

```

hKernel[hSize-1], ..., hKernel[0], ...,
hKernel[hSize-1]

```

Horizontal and vertical kernels representing convolved resample (i.e., the combined separable kernels) can be computed from a convolution filter (with odd parity), a resample filter, and because the subsample factors affect resample weights, the subsample scale factors. It is the user's responsibility to provide meaningful combined kernels.

To compute the value of a pixel centered at point (x_D, y_D) in the destination image, apply the combined kernel to the source image by aligning the kernel's geometric center to the backward mapped point (x_S, y_S) in the source image. In the cases that it can not be exactly on top of point (x_S, y_S) , the kernel's center should be half-pixel right and/or below that point. When this is done in a separable manner, the centers of horizontal and vertical kernels should align with x_S and y_S , respectively.

mllib_ImageFilteredSubsample(3MLIB)

The combination of subsampling and filtering has performance benefits over sequential function usage in part due to the symmetry constraints imposed by only allowing integer parameters for scaling and only allowing separable symmetric filters.

PARAMETERS The function takes the following arguments:

<i>dst</i>	Pointer to destination image.
<i>src</i>	Pointer to source image.
<i>scaleX</i>	The x scale factor of subsampling.
<i>scaleY</i>	The y scale factor of subsampling.
<i>transX</i>	The x translation.
<i>transY</i>	The y translation.
<i>hKernel</i>	Pointer to the compact form of horizontal kernel.
<i>vKernel</i>	Pointer to the compact form of vertical kernel.
<i>hSize</i>	Size of array <i>hKernel</i> .
<i>vSize</i>	Size of array <i>vKernel</i> .
<i>hParity</i>	Parity of horizontal kernel (0: even, 1: odd).
<i>vParity</i>	Parity of vertical kernel (0: even, 1: odd).
<i>edge</i>	Type of edge condition. It can be one of the following: <code>MLIB_EDGE_DST_NO_WRITE</code>

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_ImageSubsampleAverage\(3MLIB\)](#), [mllib_ImageZoomTranslate\(3MLIB\)](#), [mllib_ImageZoomTranslate_Fp\(3MLIB\)](#), [attributes\(5\)](#)

NAME mllib_ImageFlipAntiDiag – anti-diagonal flip

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageFlipAntiDiag(mllib_image *dst, const
mllib_image *src);
```

DESCRIPTION The mllib_ImageFlipAntiDiag() function flips an image on the anti-diagonal axis. The width and height of the destination image can be different from the width and height of the source image. The center of the source image is mapped to the center of the destination image. The data type of the images can be MLIB_BIT, MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, or MLIB_INT.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageFlipAntiDiag_Fp(3MLIB), mllib_ImageFlipMainDiag(3MLIB), mllib_ImageFlipMainDiag_Fp(3MLIB), mllib_ImageFlipX(3MLIB), mllib_ImageFlipX_Fp(3MLIB), mllib_ImageFlipY(3MLIB), mllib_ImageFlipY_Fp(3MLIB), mllib_ImageRotate90(3MLIB), mllib_ImageRotate90_Fp(3MLIB), mllib_ImageRotate180(3MLIB), mllib_ImageRotate180_Fp(3MLIB), mllib_ImageRotate270(3MLIB), mllib_ImageRotate270_Fp(3MLIB), attributes(5)

mllib_ImageFlipAntiDiag_Fp(3MLIB)

NAME | mllib_ImageFlipAntiDiag_Fp – anti-diagonal flip

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
mllib_status mllib_ImageFlipAntiDiag_Fp(mllib_image *dst, const  
mllib_image *src);
```

DESCRIPTION | The mllib_ImageFlipAntiDiag_Fp() function flips a floating-point image on the anti-diagonal axis.

The width and height of the destination image can be different from the width and height of the source image. The center of the source image is mapped to the center of the destination image.

PARAMETERS | The function takes the following arguments:

dst | Pointer to destination image.

src | Pointer to source image.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_ImageFlipAntiDiag(3MLIB), mllib_ImageFlipMainDiag(3MLIB), mllib_ImageFlipMainDiag_Fp(3MLIB), mllib_ImageFlipX(3MLIB), mllib_ImageFlipX_Fp(3MLIB), mllib_ImageFlipY(3MLIB), mllib_ImageFlipY_Fp(3MLIB), mllib_ImageRotate90(3MLIB), mllib_ImageRotate90_Fp(3MLIB), mllib_ImageRotate180(3MLIB), mllib_ImageRotate180_Fp(3MLIB), mllib_ImageRotate270(3MLIB), mllib_ImageRotate270_Fp(3MLIB), attributes(5)

NAME mllib_ImageFlipMainDiag – main diagonal flip

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageFlipMainDiag(mllib_image *dst, const
mllib_image *src);
```

DESCRIPTION The mllib_ImageFlipMainDiag() function flips an image on the main diagonal.

The width and height of the destination image can be different from the width and height of the source image. The center of the source image is mapped to the center of the destination image.

The data type of the images can be MLIB_BIT, MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, or MLIB_INT.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageFlipAntiDiag(3MLIB), mllib_ImageFlipAntiDiag_Fp(3MLIB), mllib_ImageFlipMainDiag_Fp(3MLIB), mllib_ImageFlipX(3MLIB), mllib_ImageFlipX_Fp(3MLIB), mllib_ImageFlipY(3MLIB), mllib_ImageFlipY_Fp(3MLIB), mllib_ImageRotate90(3MLIB), mllib_ImageRotate90_Fp(3MLIB), mllib_ImageRotate180(3MLIB), mllib_ImageRotate180_Fp(3MLIB), mllib_ImageRotate270(3MLIB), mllib_ImageRotate270_Fp(3MLIB), attributes(5)

mllib_ImageFlipMainDiag_Fp(3MLIB)

NAME | mllib_ImageFlipMainDiag_Fp – main diagonal flip

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageFlipMainDiag_Fp(mllib_image *dst, const
mllib_image *src);
```

DESCRIPTION | The mllib_ImageFlipMainDiag_Fp() function flips a floating-point image on the main diagonal.

The width and height of the destination image can be different from the width and height of the source image. The center of the source image is mapped to the center of the destination image.

PARAMETERS | The function takes the following arguments:

dst | Pointer to destination image.

src | Pointer to source image.

RETURN VALUES | The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_ImageFlipAntiDiag(3MLIB), mllib_ImageFlipAntiDiag_Fp(3MLIB), mllib_ImageFlipMainDiag(3MLIB), mllib_ImageFlipX(3MLIB), mllib_ImageFlipX_Fp(3MLIB), mllib_ImageFlipY(3MLIB), mllib_ImageFlipY_Fp(3MLIB), mllib_ImageRotate90(3MLIB), mllib_ImageRotate90_Fp(3MLIB), mllib_ImageRotate180(3MLIB), mllib_ImageRotate180_Fp(3MLIB), mllib_ImageRotate270(3MLIB), mllib_ImageRotate270_Fp(3MLIB), attributes(5)

NAME	mllib_ImageFlipX – X-axis flip
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageFlipX(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>);</pre>
DESCRIPTION	<p>The <code>mllib_ImageFlipX()</code> function flips an image on its X axis.</p> <p>The width and height of the destination image can be different from the width and height of the source image. The center of the source image is mapped to the center of the destination image.</p> <p>The data type of the images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageFlipAntiDiag(3MLIB)`, `mllib_ImageFlipAntiDiag_Fp(3MLIB)`, `mllib_ImageFlipMainDiag(3MLIB)`, `mllib_ImageFlipMainDiag_Fp(3MLIB)`, `mllib_ImageFlipX_Fp(3MLIB)`, `mllib_ImageFlipY(3MLIB)`, `mllib_ImageFlipY_Fp(3MLIB)`, `mllib_ImageRotate90(3MLIB)`, `mllib_ImageRotate90_Fp(3MLIB)`, `mllib_ImageRotate180(3MLIB)`, `mllib_ImageRotate180_Fp(3MLIB)`, `mllib_ImageRotate270(3MLIB)`, `mllib_ImageRotate270_Fp(3MLIB)`, `attributes(5)`

mllib_ImageFlipX_Fp(3MLIB)

NAME | mllib_ImageFlipX_Fp – X-axis flip

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>
```



```
mllib_status mllib_ImageFlipX_Fp(mllib_image *dst, const mllib_image  
*src);
```

DESCRIPTION | The `mllib_ImageFlipX_Fp()` function flips a floating-point image on its X axis.

The width and height of the destination image can be different from the width and height of the source image. The center of the source image is mapped to the center of the destination image.

PARAMETERS | The function takes the following arguments:

dst | Pointer to destination image.

src | Pointer to source image.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageFlipAntiDiag(3MLIB)`, `mllib_ImageFlipAntiDiag_Fp(3MLIB)`, `mllib_ImageFlipMainDiag(3MLIB)`, `mllib_ImageFlipMainDiag_Fp(3MLIB)`, `mllib_ImageFlipX(3MLIB)`, `mllib_ImageFlipY(3MLIB)`, `mllib_ImageFlipY_Fp(3MLIB)`, `mllib_ImageRotate90(3MLIB)`, `mllib_ImageRotate90_Fp(3MLIB)`, `mllib_ImageRotate180(3MLIB)`, `mllib_ImageRotate180_Fp(3MLIB)`, `mllib_ImageRotate270(3MLIB)`, `mllib_ImageRotate270_Fp(3MLIB)`, `attributes(5)`

NAME	mllib_ImageFlipY – Y-axis flip
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageFlipY(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>);</pre>
DESCRIPTION	<p>The <code>mllib_ImageFlipY()</code> function flips an image on its Y axis.</p> <p>The width and height of the destination image can be different from the width and height of the source image. The center of the source image is mapped to the center of the destination image.</p> <p>The data type of the images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageFlipAntiDiag(3MLIB)`, `mllib_ImageFlipAntiDiag_Fp(3MLIB)`, `mllib_ImageFlipMainDiag(3MLIB)`, `mllib_ImageFlipMainDiag_Fp(3MLIB)`, `mllib_ImageFlipX(3MLIB)`, `mllib_ImageFlipX_Fp(3MLIB)`, `mllib_ImageFlipY_Fp(3MLIB)`, `mllib_ImageRotate90(3MLIB)`, `mllib_ImageRotate90_Fp(3MLIB)`, `mllib_ImageRotate180(3MLIB)`, `mllib_ImageRotate180_Fp(3MLIB)`, `mllib_ImageRotate270(3MLIB)`, `mllib_ImageRotate270_Fp(3MLIB)`, `attributes(5)`

mllib_ImageFlipY_Fp(3MLIB)

NAME | mllib_ImageFlipY_Fp – Y-axis flip

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageFlipY_Fp(mllib_image *dst, const mllib_image
    *src);
```

DESCRIPTION | The mllib_ImageFlipY_Fp() function flips a floating-point image on its Y axis.

The width and height of the destination image can be different from the width and height of the source image. The center of the source image is mapped to the center of the destination image.

PARAMETERS | The function takes the following arguments:

dst | Pointer to destination image.

src | Pointer to source image.

RETURN VALUES | The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_ImageFlipAntiDiag(3MLIB), mllib_ImageFlipAntiDiag_Fp(3MLIB), mllib_ImageFlipMainDiag(3MLIB), mllib_ImageFlipMainDiag_Fp(3MLIB), mllib_ImageFlipX(3MLIB), mllib_ImageFlipX_Fp(3MLIB), mllib_ImageFlipY(3MLIB), mllib_ImageRotate90(3MLIB), mllib_ImageRotate90_Fp(3MLIB), mllib_ImageRotate180(3MLIB), mllib_ImageRotate180_Fp(3MLIB), mllib_ImageRotate270(3MLIB), mllib_ImageRotate270_Fp(3MLIB), attributes(5)

NAME mllib_ImageFourierTransform – Fourier transform

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>
```

```
mllib_status mllib_ImageFourierTransform(mllib_image *dst, const
mllib_image *src, mllib_fourier_mode mode);
```

DESCRIPTION The `mllib_ImageFourierTransform()` function performs a two-dimensional Fourier transformation. The source and destination images must be the same type and the same size. The data type of the images can be `MLIB_BYTE`, `MLIB_SHORT`, `MLIB_USHORT`, `MLIB_INT`, `MLIB_FLOAT`, or `MLIB_DOUBLE`. The height and width of the images must be some positive power of 2 (but they do not have to be equal).

They can have 1 or 2 channels. If the source image has just one channel the imaginary parts are assumed to be zero. If the destination image has just one channel, then it is assumed that the imaginary parts of the output can be discarded. But in case both source and destination images are one-channel images, then `MLIB_FAILURE` is returned.

The predefined modes used in the image Fourier transform function are as follows:

Mode	Description
<code>MLIB_DFT_SCALE_NONE</code>	Forward DFT without scaling
<code>MLIB_DFT_SCALE_MXN</code>	Forward DFT with scaling of 1/(M*N)
<code>MLIB_DFT_SCALE_SQRT</code>	Forward DFT with scaling of 1/sqrt(M*N)
<code>MLIB_IDFT_SCALE_NONE</code>	Inverse DFT without scaling
<code>MLIB_IDFT_SCALE_MXN</code>	Inverse DFT with scaling of 1/(M*N)
<code>MLIB_IDFT_SCALE_SQRT</code>	Inverse DFT with scaling of 1/sqrt(M*N)

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.
src Pointer to source image.
mode Mode of the transform.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

mllib_ImageFourierTransform(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	MT-Safe

SEE ALSO [attributes\(5\)](#)

NAME mllib_ImageGetBitOffset – get bitoffset

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_s32 mllib_ImageGetBitOffset(const mllib_image *img);
```

DESCRIPTION A query function that returns the bitoffset public field of a mediaLib image structure. The data type of the image can be MLIB_BIT, MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, MLIB_INT, MLIB_FLOAT, or MLIB_DOUBLE.

PARAMETERS The function takes the following arguments:

img Pointer to a mediaLib image structure.

RETURN VALUES The function returns the offset, in terms of bits, of an image from the beginning of the data buffer to the first pixel.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageGetChannels(3MLIB)`, `mllib_ImageGetData(3MLIB)`, `mllib_ImageGetFlags(3MLIB)`, `mllib_ImageGetHeight(3MLIB)`, `mllib_ImageGetPaddings(3MLIB)`, `mllib_ImageGetStride(3MLIB)`, `mllib_ImageGetType(3MLIB)`, `mllib_ImageGetWidth(3MLIB)`, `attributes(5)`

mllib_ImageGetChannels(3MLIB)

NAME | mllib_ImageGetChannels – get channels

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
mllib_s32 mllib_ImageGetChannels(const mllib_image *img);
```

DESCRIPTION | A query function that returns the channels public field of a mediaLib image structure. The data type of the image can be MLIB_BIT, MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, MLIB_INT, MLIB_FLOAT, or MLIB_DOUBLE.

PARAMETERS | The function takes the following arguments:
img Pointer to a mediaLib image structure.

RETURN VALUES | The function returns the number of channels in an image.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageGetBitOffset(3MLIB)`, `mllib_ImageGetData(3MLIB)`, `mllib_ImageGetFlags(3MLIB)`, `mllib_ImageGetHeight(3MLIB)`, `mllib_ImageGetPaddings(3MLIB)`, `mllib_ImageGetStride(3MLIB)`, `mllib_ImageGetType(3MLIB)`, `mllib_ImageGetWidth(3MLIB)`, `attributes(5)`

NAME mllib_ImageGetData – get data

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void *mllib_ImageGetData(const mllib_image *img);
```

DESCRIPTION The mllib_ImageGetData() function returns the data public field of a mediaLib image structure. The data type of the image can be MLIB_BIT, MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, MLIB_INT, MLIB_FLOAT, or MLIB_DOUBLE.

PARAMETERS The function takes the following arguments:

img Pointer to source image.

RETURN VALUES The function returns a pointer to the image data.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageGetBitOffset(3MLIB), mllib_ImageGetChannels(3MLIB), mllib_ImageGetFlags(3MLIB), mllib_ImageGetHeight(3MLIB), mllib_ImageGetPaddings(3MLIB), mllib_ImageGetStride(3MLIB), mllib_ImageGetType(3MLIB), mllib_ImageGetWidth(3MLIB), attributes(5)

mllib_ImageGetFlags(3MLIB)

NAME | mllib_ImageGetFlags – get flags

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
mllib_s32 mllib_ImageGetFlags(const mllib_image *img);
```

DESCRIPTION | The mllib_ImageGetFlags() function returns the attribute flags of an image. The data type of the image can be MLIB_BIT, MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, MLIB_INT, MLIB_FLOAT, or MLIB_DOUBLE.

PARAMETERS | The function takes the following arguments:
img Pointer to source image.

RETURN VALUES | The function returns the value of the attribute flags.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_ImageGetBitOffset(3MLIB), mllib_ImageGetChannels(3MLIB), mllib_ImageGetData(3MLIB), mllib_ImageGetHeight(3MLIB), mllib_ImageGetPaddings(3MLIB), mllib_ImageGetStride(3MLIB), mllib_ImageGetType(3MLIB), mllib_ImageGetWidth(3MLIB), attributes(5)

NAME mllib_ImageGetFormat – get format

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_format mllib_ImageGetFormat(const mllib_image *img);
```

DESCRIPTION A query function that returns the format public field of a mllib_image structure. The data type of the image can be MLIB_BIT, MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, MLIB_INT, MLIB_FLOAT, or MLIB_DOUBLE.

PARAMETERS The function takes the following arguments:
img Pointer to a mediaLib image structure.

RETURN VALUES The function returns the value of the format of an image.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageSetFormat(3MLIB), mllib_ImageGetBitOffset(3MLIB), mllib_ImageGetChannels(3MLIB), mllib_ImageGetData(3MLIB), mllib_ImageGetFlags(3MLIB), mllib_ImageGetHeight(3MLIB), mllib_ImageGetPaddings(3MLIB), mllib_ImageGetStride(3MLIB), mllib_ImageGetType(3MLIB), mllib_ImageGetWidth(3MLIB), attributes(5)

mllib_ImageGetHeight(3MLIB)

NAME | mllib_ImageGetHeight – get height

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
mllib_s32 mllib_ImageGetHeight(const mllib_image *img);
```

DESCRIPTION | A query function that returns the height public field of a mediaLib image structure. The data type of the image can be MLIB_BIT, MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, MLIB_INT, MLIB_FLOAT, or MLIB_DOUBLE.

PARAMETERS | The function takes the following arguments:
img Pointer to source image.

RETURN VALUES | The function returns the value of the height (in pixels) of an image.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageGetBitOffset(3MLIB)`, `mllib_ImageGetChannels(3MLIB)`, `mllib_ImageGetData(3MLIB)`, `mllib_ImageGetFlags(3MLIB)`, `mllib_ImageGetPaddings(3MLIB)`, `mllib_ImageGetStride(3MLIB)`, `mllib_ImageGetType(3MLIB)`, `mllib_ImageGetWidth(3MLIB)`, `attributes(5)`

NAME	mlib_ImageGetPaddings – get paddings
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmlib [<i>library...</i>] #include <mlib.h> mlib_u8 *mlib_ImageGetPaddings(const mlib_image *img);</pre>
DESCRIPTION	A query function that returns the borders public field of a mediaLib image structure. The data type of the image can be MLIB_BIT, MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, MLIB_INT, MLIB_FLOAT, or MLIB_DOUBLE.
PARAMETERS	The function takes the following arguments: <i>img</i> Pointer to a mediaLib image structure.
RETURN VALUES	The function returns a pointer to the image paddings. paddings[0] holds leftPadding; paddings[1] holds topPadding; paddings[2] holds rightPadding; paddings[3] holds bottomPadding.
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mlib_ImageGetBitOffset(3MLIB), mlib_ImageGetChannels(3MLIB), mlib_ImageGetData(3MLIB), mlib_ImageGetFlags(3MLIB), mlib_ImageGetHeight(3MLIB), mlib_ImageGetStride(3MLIB), mlib_ImageGetType(3MLIB), mlib_ImageGetWidth(3MLIB), mlib_ImageSetPaddings(3MLIB), attributes(5)

mllib_ImageGetStride(3MLIB)

NAME | mllib_ImageGetStride – get stride

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
mllib_s32 mllib_ImageGetStride(const mllib_image *img);
```

DESCRIPTION | A query function that returns the stride public field of a mediaLib image structure. The data type of the image can be MLIB_BIT, MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, MLIB_INT, MLIB_FLOAT, or MLIB_DOUBLE.

PARAMETERS | The function takes the following arguments:
img Pointer to source image.

RETURN VALUES | The function returns the value of the stride (in bytes) of an image.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageGetBitOffset(3MLIB)`, `mllib_ImageGetChannels(3MLIB)`, `mllib_ImageGetData(3MLIB)`, `mllib_ImageGetFlags(3MLIB)`, `mllib_ImageGetHeight(3MLIB)`, `mllib_ImageGetPaddings(3MLIB)`, `mllib_ImageGetType(3MLIB)`, `mllib_ImageGetWidth(3MLIB)`, `attributes(5)`

NAME mllib_ImageGetType – get type

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_type mllib_ImageGetType(const mllib_image *img);
```

DESCRIPTION A query function that returns the type public field of a mediaLib image structure. The data type of the image can be MLIB_BIT, MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, MLIB_INT, MLIB_FLOAT, or MLIB_DOUBLE.

PARAMETERS The function takes the following arguments:

img Pointer to source image.

RETURN VALUES The function returns the value of the type of an image.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageGetBitOffset(3MLIB)`, `mllib_ImageGetChannels(3MLIB)`, `mllib_ImageGetData(3MLIB)`, `mllib_ImageGetFlags(3MLIB)`, `mllib_ImageGetHeight(3MLIB)`, `mllib_ImageGetPaddings(3MLIB)`, `mllib_ImageGetStride(3MLIB)`, `mllib_ImageGetWidth(3MLIB)`, `attributes(5)`

mllib_ImageGetWidth(3MLIB)

NAME | mllib_ImageGetWidth – get width

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
mllib_s32 mllib_ImageGetWidth(const mllib_image *img);
```

DESCRIPTION | A query function that returns the width public field of a mediaLib image structure. The data type of the image can be MLIB_BIT, MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, MLIB_INT, MLIB_FLOAT, or MLIB_DOUBLE.

PARAMETERS | The function takes the following arguments:
img Pointer to source image.

RETURN VALUES | The function returns the value of the width of an image.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_ImageGetBitOffset\(3MLIB\)](#), [mllib_ImageGetChannels\(3MLIB\)](#), [mllib_ImageGetData\(3MLIB\)](#), [mllib_ImageGetFlags\(3MLIB\)](#), [mllib_ImageGetHeight\(3MLIB\)](#), [mllib_ImageGetPaddings\(3MLIB\)](#), [mllib_ImageGetStride\(3MLIB\)](#), [mllib_ImageGetType\(3MLIB\)](#), [attributes\(5\)](#)

NAME mllib_ImageGradient3x3 – 3x3 gradient filter

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageGradient3x3(mllib_image *dst, const
    mllib_image *src, const mllib_d64 *hmask, const mllib_d64 *vmask,
    mllib_s32 cmask, mllib_edge edge);
```

DESCRIPTION The `mllib_ImageGradient3x3()` function performs edge detection by computing the magnitude of the image gradient vector in two orthogonal directions using 3x3 gradient filtering.

It uses the following equation:

$$dst[x][y][i] = (SH(x,y,i)**2 + SV(x,y,i)**2)**0.5$$

where `SH()` and `SV()` are the horizontal and vertical gradient images generated from the corresponding channel of the source image by correlating it with the supplied orthogonal (horizontal and vertical) gradient masks.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

hmask Pointer to horizontal mask in row-major order.

vmask Pointer to vertical mask in row-major order.

cmask Channel mask to indicate the channels to be convolved, each bit of which represents a channel in the image. The channels corresponding to 1 bits are those to be processed. For a single channel image, the channel mask is ignored.

edge Type of edge condition. It can be one of the following:

```
MLIB_EDGE_DST_NO_WRITE
MLIB_EDGE_DS_FILL_ZERO
MLIB_EDGE_DST_COPY_SRC
MLIB_EDGE_SR_EXTEND
```

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

`mllib_ImageGradient3x3(3MLIB)`

SEE ALSO `mllib_ImageGradient3x3_Fp(3MLIB)`, `mllib_ImageGradientMxN(3MLIB)`,
`mllib_ImageGradientMxN_Fp(3MLIB)`, `attributes(5)`

NAME mllib_ImageGradient3x3_Fp – 3x3 gradient filter

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageGradient3x3_Fp(mllib_image *dst, const
    mllib_image *src, const mllib_d64 *hmask, const mllib_d64 *vmask,
    mllib_s32 cmask, mllib_edge edge);
```

DESCRIPTION The `mllib_ImageGradient3x3_Fp()` function performs floating-point edge detection by computing the magnitude of the image gradient vector in two orthogonal directions using 3x3 gradient filtering.

It uses the following equation:

$$dst[x][y][i] = (SH(x,y,i)**2 + SV(x,y,i)**2)**0.5$$

where `SH()` and `SV()` are the horizontal and vertical gradient images generated from the corresponding channel of the source image by correlating it with the supplied orthogonal (horizontal and vertical) gradient masks.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

hmask Pointer to horizontal mask in row-major order.

vmask Pointer to vertical mask in row-major order.

cmask Channel mask to indicate the channels to be convolved, each bit of which represents a channel in the image. The channels corresponding to 1 bits are those to be processed. For a single channel image, the channel mask is ignored.

edge Type of edge condition. It can be one of the following:

```
MLIB_EDGE_DST_NO_WRITE
MLIB_EDGE_DS_FILL_ZERO
MLIB_EDGE_DST_COPY_SRC
MLIB_EDGE_SR_EXTEND
```

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

`mlib_ImageGradient3x3_Fp(3MLIB)`

SEE ALSO | `mlib_ImageGradient3x3(3MLIB)`, `mlib_ImageGradientMxN(3MLIB)`,
`mlib_ImageGradientMxN_Fp(3MLIB)`, `attributes(5)`

NAME	mllib_ImageGradientMxN – MxN gradient filter
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageGradientMxN(mllib_image *dst, const mllib_image *src, const mllib_d64 *hmask, const mllib_d64 *vmask, mllib_s32 m, mllib_s32 n, mllib_s32 dm, mllib_s32 dn, mllib_s32 cmask, mllib_edge edge);</pre>
DESCRIPTION	<p>The <code>mllib_ImageGradientMxN()</code> function performs edge detection by computing the magnitude of the image gradient vector in two orthogonal directions using MxN gradient filtering.</p> <p>It uses the following equation:</p> $dst[x][y][i] = (SH(x,y,i)**2 + SV(x,y,i)**2)**0.5$ <p>where <code>SH()</code> and <code>SV()</code> are the horizontal and vertical gradient images generated from the corresponding channel of the source image by correlating it with the supplied orthogonal (horizontal and vertical) gradient masks.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>hmask</i> Pointer to horizontal mask in row-major order.</p> <p><i>vmask</i> Pointer to vertical mask in row-major order.</p> <p><i>m</i> Width of the convolution kernel. $m > 1$.</p> <p><i>n</i> Height of the convolution kernel. $n > 1$.</p> <p><i>dm</i> X coordinate of the key element in the convolution kernel. $0 \leq dm < m$.</p> <p><i>dn</i> Y coordinate of the key element in the convolution kernel. $0 \leq dn < n$.</p> <p><i>cmask</i> Channel mask to indicate the channels to be convolved, each bit of which represents a channel in the image. The channels corresponding to 1 bits are those to be processed. For a single channel image, the channel mask is ignored.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DS_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SR_EXTEND</pre>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .

mllib_ImageGradientMxN(3MLIB)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageGradientMxN_Fp(3MLIB)`, `mllib_ImageGradient3x3(3MLIB)`,
`mllib_ImageGradient3x3_Fp(3MLIB)`, `attributes(5)`

NAME	mllib_ImageGradientMxN_Fp – MxN gradient filter
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageGradientMxN_Fp(mllib_image *dst, const mllib_image *src, const mllib_d64 *hmask, const mllib_d64 *vmask, mllib_s32 m, mllib_s32 n, mllib_s32 dm, mllib_s32 dn, mllib_s32 cmask, mllib_edge edge);</pre>
DESCRIPTION	<p>The <code>mllib_ImageGradientMxN_Fp()</code> function performs floating-point edge detection by computing the magnitude of the image gradient vector in two orthogonal directions using MxN gradient filtering.</p> <p>It uses the following equation:</p> $dst[x][y][i] = (SH(x,y,i)**2 + SV(x,y,i)**2)**0.5$ <p>where <code>SH()</code> and <code>SV()</code> are the horizontal and vertical gradient images generated from the corresponding channel of the source image by correlating it with the supplied orthogonal (horizontal and vertical) gradient masks.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>hmask</i> Pointer to horizontal mask in row-major order.</p> <p><i>vmask</i> Pointer to vertical mask in row-major order.</p> <p><i>m</i> Width of the convolution kernel. $m > 1$.</p> <p><i>n</i> Height of the convolution kernel. $n > 1$.</p> <p><i>dm</i> X coordinate of the key element in the convolution kernel. $0 \leq dm < m$.</p> <p><i>dn</i> Y coordinate of the key element in the convolution kernel. $0 \leq dn < n$.</p> <p><i>cmask</i> Channel mask to indicate the channels to be convolved, each bit of which represents a channel in the image. The channels corresponding to 1 bits are those to be processed. For a single channel image, the channel mask is ignored.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DS_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SR_EXTEND</pre>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .

`mllib_ImageGradientMxN_Fp(3MLIB)`

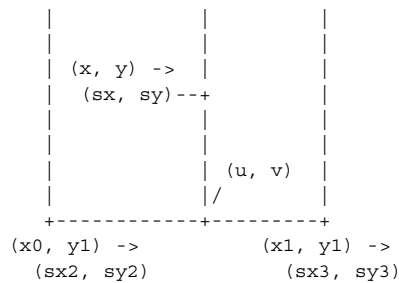
ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageGradientMxN(3MLIB)`, `mllib_ImageGradient3x3(3MLIB)`,
`mllib_ImageGradient3x3_Fp(3MLIB)`, `attributes(5)`

NAME	mllib_ImageGridWarp – grid-based image warp
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageGridWarp(mllib_image *dst, const mllib_image *src, const mllib_f32 *xWarpPos, const mllib_f32 *yWarpPos, mllib_d64 postShiftX, mllib_d64 postShiftY, mllib_s32 xStart, mllib_s32 xStep, mllib_s32 xNumCells, mllib_s32 yStart, mllib_s32 yStep, mllib_s32 yNumCells, mllib_filter filter, mllib_edge edge);</pre>
DESCRIPTION	<p>The <code>mllib_ImageGridWarp()</code> function performs a regular grid-based image warp. The images must have the same type, and the same number of channels. The images can have 1, 2, 3, or 4 channels. The data type of the images can be <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>. The two images may have different sizes.</p> <p>The image pixels are assumed to be centered at .5 coordinate points. For example, the upper-left corner pixel of an image is located at (0.5, 0.5).</p> <p>For each pixel in the destination image, its center point D is, first, backward mapped to a point S in the source image; then the source pixels with their centers surrounding point S are selected to do one of the interpolations specified by the filter parameter to generate the pixel value for point D.</p> <p>The mapping from destination pixels to source positions is described by bilinear interpolation between a rectilinear grid of points with known mappings.</p> <p>Given a destination pixel coordinate (x, y) that lies within a cell having corners at (x0, y0), (x1, y0), (x0, y1) and (x1, y1), with source coordinates defined at each respective corner equal to (sx0, sy0), (sx1, sy1), (sx2, sy2) and (sx3, sy3), the source position (sx, sy) that maps onto (x, y) is given by the formulas:</p> <pre>xfrac = (x - x0) / (x1 - x0) yfrac = (y - y0) / (y1 - y0) s = sx0 + (sx1 - sx0) * xfrac t = sy0 + (sy1 - sy0) * xfrac u = sx2 + (sx3 - sx2) * yfrac v = sy2 + (sy3 - sy2) * yfrac sx = s + (u - s) * yfrac - postShiftX sy = t + (v - t) * yfrac - postShiftY</pre> <p>In other words, the source x and y values are interpolated horizontally along the top and bottom edges of the grid cell, and the results are interpolated vertically:</p> <pre>(x0, y0) -> (x1, y0) -> (sx0, sy0) (sx1, sy1) +-----+-----+ / (s, t) </pre>

mllib_ImageGridWarp(3MLIB)



The results of above interpolation are shifted by `(-postShiftX, -postShiftY)` to produce the source pixel coordinates.

The destination pixels that lie outside of any grid cells are kept intact. The grid is defined by a set of equal-sized cells. The grid starts at `(xStart, yStart)`. Each cell has width equal to `xStep` and height equal to `yStep`, and there are `xNumCells` cells horizontally and `yNumCells` cells vertically.

The degree of warping within each cell is defined by the values in `xWarpPos` and `yWarpPos` parameters. Each of these parameters must contain $(xNumCells + 1) * (yNumCells + 1)$ values, which, respectively, contain the source X and source Y coordinates that map to the upper-left corner of each cell in the destination image. The cells are enumerated in row-major order. That is, all the grid points along a row are enumerated first, then the grid points for the next row are enumerated, and so on.

For example, suppose `xNumCells` is equal to 2 and `yNumCells` is equal to 1. Then the order of the data in the `xWarpPos` would be:

```
x00, x10, x20, x01, x11, x21
```

and in the `yWarpPos`:

```
y00, y10, y20, y01, y11, y21
```

for a total of $(2 + 1) * (1 + 1) = 6$ elements in each table.

PARAMETERS

The function takes the following arguments:

<i>dst</i>	Pointer to destination image.
<i>src</i>	Pointer to source image.
<i>xWarpPos</i>	A float array of length $(xNumCells + 1) * (yNumCells + 1)$ containing horizontal warp positions at the grid points, in row-major order.
<i>yWarpPos</i>	A float array of length $(xNumCells + 1) * (yNumCells + 1)$ containing vertical warp positions at the grid points, in row-major order.
<i>postShiftX</i>	The displacement to apply to source X positions.
<i>postShiftY</i>	The displacement to apply to source Y positions.

mllib_ImageGridWarp(3MLIB)

xStart The minimum X coordinate of the grid.

xStep The horizontal spacing between grid cells.

xNumCells The number of grid cell columns.

yStart The minimum Y coordinate of the grid.

yStep The vertical spacing between grid cells.

yNumCells The number of grid cell rows.

filter Type of resampling filter. It can be one of the following:
 MLIB_NEAREST
 MLIB_BILINEAR
 MLIB_BICUBIC
 MLIB_BICUBIC2

edge Type of edge condition. It can be one of the following:
 MLIB_EDGE_DST_NO_WRITE
 MLIB_EDGE_SRC_PADDED

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

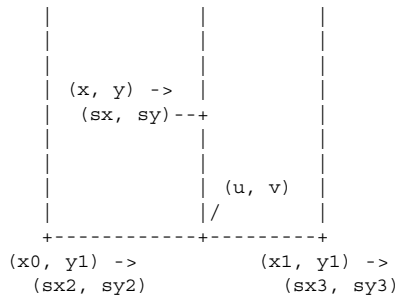
ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageGridWarp_Fp(3MLIB), mllib_ImageGridWarpTable(3MLIB), mllib_ImageGridWarpTable_Fp(3MLIB), attributes(5)

mllib_ImageGridWarp_Fp(3MLIB)

NAME	mllib_ImageGridWarp_Fp – grid-based image warp
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageGridWarp_Fp(mllib_image *dst, const mllib_image *src, const mllib_f32 *xWarpPos, const mllib_f32 *yWarpPos, mllib_d64 postShiftX, mllib_d64 postShiftY, mllib_s32 xStart, mllib_s32 xStep, mllib_s32 xNumCells, mllib_s32 yStart, mllib_s32 yStep, mllib_s32 yNumCells, mllib_filter filter, mllib_edge edge);</pre>
DESCRIPTION	<p>The <code>mllib_ImageGridWarp_Fp()</code> function performs a regular grid-based image warp on a floating-point image. The images must have the same type, and the same number of channels. The images can have 1, 2, 3, or 4 channels. The data type of the images can be <code>MLIB_FLOAT</code> or <code>MLIB_DOUBLE</code>. The two images may have different sizes.</p> <p>The image pixels are assumed to be centered at .5 coordinate points. For example, the upper-left corner pixel of an image is located at (0.5, 0.5).</p> <p>For each pixel in the destination image, its center point D is, first, backward mapped to a point S in the source image; then the source pixels with their centers surrounding point S are selected to do one of the interpolations specified by the filter parameter to generate the pixel value for point D.</p> <p>The mapping from destination pixels to source positions is described by bilinear interpolation between a rectilinear grid of points with known mappings.</p> <p>Given a destination pixel coordinate (x, y) that lies within a cell having corners at (x0, y0), (x1, y0), (x0, y1) and (x1, y1), with source coordinates defined at each respective corner equal to (sx0, sy0), (sx1, sy1), (sx2, sy2) and (sx3, sy3), the source position (sx, sy) that maps onto (x, y) is given by the formulas:</p> <pre>xfrac = (x - x0)/(x1 - x0) yfrac = (y - y0)/(y1 - y0) s = sx0 + (sx1 - sx0)*xfrac t = sy0 + (sy1 - sy0)*xfrac u = sx2 + (sx3 - sx2)*xfrac v = sy2 + (sy3 - sy2)*xfrac sx = s + (u - s)*yfrac - postShiftX sy = t + (v - t)*yfrac - postShiftY</pre> <p>In other words, the source x and y values are interpolated horizontally along the top and bottom edges of the grid cell, and the results are interpolated vertically:</p> <pre>(x0, y0) -> (x1, y0) -> (sx0, sy0) (sx1, sy1) +-----+-----+ / (s, t) +-----+-----+</pre>



The results of above interpolation are shifted by $(-postShiftX, -postShiftY)$ to produce the source pixel coordinates.

The destination pixels that lie outside of any grid cells are kept intact. The grid is defined by a set of equal-sized cells. The grid starts at $(xStart, yStart)$. Each cell has width equal to $xStep$ and height equal to $yStep$, and there are $xNumCells$ cells horizontally and $yNumCells$ cells vertically.

The degree of warping within each cell is defined by the values in $xWarpPos$ and $yWarpPos$ parameters. Each of these parameters must contain $(xNumCells + 1) * (yNumCells + 1)$ values, which, respectively, contain the source X and source Y coordinates that map to the upper-left corner of each cell in the destination image. The cells are enumerated in row-major order. That is, all the grid points along a row are enumerated first, then the grid points for the next row are enumerated, and so on.

For example, suppose $xNumCells$ is equal to 2 and $yNumCells$ is equal to 1. Then the order of the data in the $xWarpPos$ would be:

`x00, x10, x20, x01, x11, x21`

and in the $yWarpPos$:

`y00, y10, y20, y01, y11, y21`

for a total of $(2 + 1) * (1 + 1) = 6$ elements in each table.

PARAMETERS

The function takes the following arguments:

- dst* Pointer to destination image.
- src* Pointer to source image.
- xWarpPos* A float array of length $(xNumCells + 1) * (yNumCells + 1)$ containing horizontal warp positions at the grid points, in row-major order.
- yWarpPos* A float array of length $(xNumCells + 1) * (yNumCells + 1)$ containing vertical warp positions at the grid points, in row-major order.
- postShiftX* The displacement to apply to source X positions.
- postShiftY* The displacement to apply to source Y positions.

mllib_ImageGridWarp_Fp(3MLIB)

<i>xStart</i>	The minimum X coordinate of the grid.
<i>xStep</i>	The horizontal spacing between grid cells.
<i>xNumCells</i>	The number of grid cell columns.
<i>yStart</i>	The minimum Y coordinate of the grid.
<i>yStep</i>	The vertical spacing between grid cells.
<i>yNumCells</i>	The number of grid cell rows.
<i>filter</i>	Type of resampling filter. It can be one of the following: MLIB_NEAREST MLIB_BILINEAR MLIB_BICUBIC MLIB_BICUBIC2
<i>edge</i>	Type of edge condition. It can be one of the following: MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_SRC_PADDED

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageGridWarp(3MLIB)`, `mllib_ImageGridWarpTable(3MLIB)`, `mllib_ImageGridWarpTable_Fp(3MLIB)`, `attributes(5)`

NAME mllib_ImageGridWarpTable – grid-based image warp with table-driven interpolation

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageGridWarpTable(mllib_image *dst, const
    mllib_image *src, const mllib_f32 *xWarpPos, const mllib_f32
    *yWarpPos, mllib_d64 postShiftX, mllib_d64 postShiftY, mllib_s32 xStart,
    mllib_s32 xStep, mllib_s32 xNumCells, mllib_s32 yStart, mllib_s32
    yStep, mllib_s32 yNumCells, const void *interp_table, mllib_edge
    edge);
```

DESCRIPTION The mllib_ImageGridWarpTable() function performs a regular grid-based image warp with table-driven interpolation. The images must have the same type, and the same number of channels. The images can have 1, 2, 3, or 4 channels. The data type of the images can be MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, or MLIB_INT. The two images may have different sizes.

The image pixels are assumed to be centered at .5 coordinate points. For example, the upper-left corner pixel of an image is located at (0.5, 0.5).

For each pixel in the destination image, its center point D is, first, backward mapped to a point S in the source image; then the source pixels with their centers surrounding point S are selected to do one of the interpolations specified by the filter parameter to generate the pixel value for point D.

The mapping from destination pixels to source positions is described by bilinear interpolation between a rectilinear grid of points with known mappings.

Given a destination pixel coordinate (x, y) that lies within a cell having corners at (x0, y0), (x1, y0), (x0, y1) and (x1, y1), with source coordinates defined at each respective corner equal to (sx0, sy0), (sx1, sy1), (sx2, sy2) and (sx3, sy3), the source position (sx, sy) that maps onto (x, y) is given by the formulas:

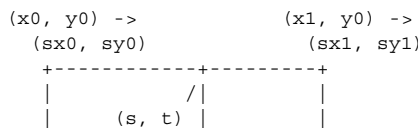
```
xfrac = (x - x0) / (x1 - x0)
yfrac = (y - y0) / (y1 - y0)

s = sx0 + (sx1 - sx0) * xfrac
t = sy0 + (sy1 - sy0) * xfrac

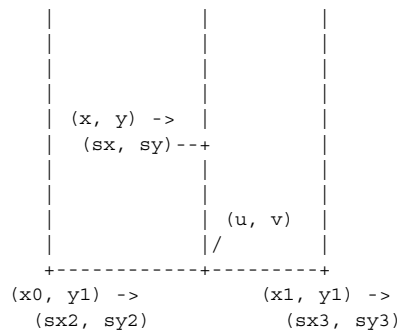
u = sx2 + (sx3 - sx2) * xfrac
v = sy2 + (sy3 - sy2) * xfrac

sx = s + (u - s) * yfrac - postShiftX
sy = t + (v - t) * yfrac - postShiftY
```

In other words, the source x and y values are interpolated horizontally along the top and bottom edges of the grid cell, and the results are interpolated vertically:



mllib_ImageGridWarpTable(3MLIB)



The results of above interpolation are shifted by $(-postShiftX, -postShiftY)$ to produce the source pixel coordinates.

The destination pixels that lie outside of any grid cells are kept intact. The grid is defined by a set of equal-sized cells. The grid starts at $(xStart, yStart)$. Each cell has width equal to $xStep$ and height equal to $yStep$, and there are $xNumCells$ cells horizontally and $yNumCells$ cells vertically.

The degree of warping within each cell is defined by the values in $xWarpPos$ and $yWarpPos$ parameters. Each of these parameters must contain $(xNumCells + 1) * (yNumCells + 1)$ values, which, respectively, contain the source X and source Y coordinates that map to the upper-left corner of each cell in the destination image. The cells are enumerated in row-major order. That is, all the grid points along a row are enumerated first, then the grid points for the next row are enumerated, and so on.

For example, suppose $xNumCells$ is equal to 2 and $yNumCells$ is equal to 1. Then the order of the data in the $xWarpPos$ would be:

```
x00, x10, x20, x01, x11, x21
```

and in the $yWarpPos$:

```
y00, y10, y20, y01, y11, y21
```

for a total of $(2 + 1) * (1 + 1) = 6$ elements in each table.

PARAMETERS

The function takes the following arguments:

<i>dst</i>	Pointer to destination image.
<i>src</i>	Pointer to source image.
<i>xWarpPos</i>	A float array of length $(xNumCells + 1) * (yNumCells + 1)$ containing horizontal warp positions at the grid points, in row-major order.
<i>yWarpPos</i>	A float array of length $(xNumCells + 1) * (yNumCells + 1)$ containing vertical warp positions at the grid points, in row-major order.
<i>postShiftX</i>	The displacement to apply to source X positions.

mllib_ImageGridWarpTable(3MLIB)

postShiftY The displacement to apply to source Y positions.

xStart The minimum X coordinate of the grid.

xStep The horizontal spacing between grid cells.

xNumCells The number of grid cell columns.

yStart The minimum Y coordinate of the grid.

yStep The vertical spacing between grid cells.

yNumCells The number of grid cell rows.

interp_table Pointer to an interpolation table. The table is created by the `mllib_ImageInterpTableCreate()` function.

edge Type of edge condition. It can be one of the following:
 MLIB_EDGE_DST_NO_WRITE
 MLIB_EDGE_SRC_PADDED

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

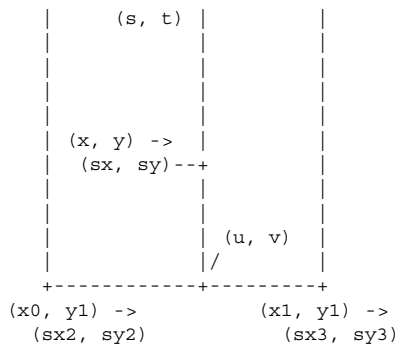
ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageInterpTableCreate(3MLIB)`,
`mllib_ImageInterpTableDelete(3MLIB)`, `mllib_ImageGridWarp(3MLIB)`,
`mllib_ImageGridWarp_Fp(3MLIB)`, `mllib_ImageGridWarpTable_Fp(3MLIB)`,
`attributes(5)`

mllib_ImageGridWarpTable_Fp(3MLIB)

NAME	mllib_ImageGridWarpTable_Fp – grid-based image warp with table-driven interpolation
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageGridWarpTable_Fp(mllib_image *dst, const mllib_image *src, const mllib_f32 *xWarpPos, const mllib_f32 *yWarpPos, mllib_d64 postShiftX, mllib_d64 postShiftY, mllib_s32 xStart, mllib_s32 xStep, mllib_s32 xNumCells, mllib_s32 yStart, mllib_s32 yStep, mllib_s32 yNumCells, const void *interp_table, mllib_edge edge);</pre>
DESCRIPTION	<p>The <code>mllib_ImageGridWarpTable_Fp()</code> function performs a regular grid-based image warp on a floating-point image with table-driven interpolation. The images must have the same type, and the same number of channels. The images can have 1, 2, 3, or 4 channels. The data type of the images can be <code>MLIB_FLOAT</code> or <code>MLIB_DOUBLE</code>. The two images may have different sizes.</p> <p>The image pixels are assumed to be centered at .5 coordinate points. For example, the upper-left corner pixel of an image is located at (0.5, 0.5).</p> <p>For each pixel in the destination image, its center point D is, first, backward mapped to a point S in the source image; then the source pixels with their centers surrounding point S are selected to do one of the interpolations specified by the filter parameter to generate the pixel value for point D.</p> <p>The mapping from destination pixels to source positions is described by bilinear interpolation between a rectilinear grid of points with known mappings.</p> <p>Given a destination pixel coordinate (x, y) that lies within a cell having corners at (x0, y0), (x1, y0), (x0, y1) and (x1, y1), with source coordinates defined at each respective corner equal to (sx0, sy0), (sx1, sy1), (sx2, sy2) and (sx3, sy3), the source position (sx, sy) that maps onto (x, y) is given by the formulas:</p> <pre>xfrac = (x - x0)/(x1 - x0) yfrac = (y - y0)/(y1 - y0) s = sx0 + (sx1 - sx0)*xfrac t = sy0 + (sy1 - sy0)*xfrac u = sx2 + (sx3 - sx2)*xfrac v = sy2 + (sy3 - sy2)*xfrac sx = s + (u - s)*yfrac - postShiftX sy = t + (v - t)*yfrac - postShiftY</pre> <p>In other words, the source x and y values are interpolated horizontally along the top and bottom edges of the grid cell, and the results are interpolated vertically:</p> <pre> (x0, y0) -> (x1, y0) -> (sx0, sy0) (sx1, sy1) +-----+-----+ / </pre>



The results of above interpolation are shifted by $(-postShiftX, -postShiftY)$ to produce the source pixel coordinates.

The destination pixels that lie outside of any grid cells are kept intact. The grid is defined by a set of equal-sized cells. The grid starts at $(xStart, yStart)$. Each cell has width equal to $xStep$ and height equal to $yStep$, and there are $xNumCells$ cells horizontally and $yNumCells$ cells vertically.

The degree of warping within each cell is defined by the values in $xWarpPos$ and $yWarpPos$ parameters. Each of these parameters must contain $(xNumCells + 1) * (yNumCells + 1)$ values, which, respectively, contain the source X and source Y coordinates that map to the upper-left corner of each cell in the destination image. The cells are enumerated in row-major order. That is, all the grid points along a row are enumerated first, then the grid points for the next row are enumerated, and so on.

For example, suppose $xNumCells$ is equal to 2 and $yNumCells$ is equal to 1. Then the order of the data in the $xWarpPos$ would be:

$x00, x10, x20, x01, x11, x21$

and in the $yWarpPos$:

$y00, y10, y20, y01, y11, y21$

for a total of $(2 + 1) * (1 + 1) = 6$ elements in each table.

PARAMETERS

The function takes the following arguments:

<i>dst</i>	Pointer to destination image.
<i>src</i>	Pointer to source image.
<i>xWarpPos</i>	A float array of length $(xNumCells + 1) * (yNumCells + 1)$ containing horizontal warp positions at the grid points, in row-major order.
<i>yWarpPos</i>	A float array of length $(xNumCells + 1) * (yNumCells + 1)$ containing vertical warp positions at the grid points, in row-major order.
<i>postShiftX</i>	The displacement to apply to source X positions.

mllib_ImageGridWarpTable_Fp(3MLIB)

<i>postShiftY</i>	The displacement to apply to source Y positions.
<i>xStart</i>	The minimum X coordinate of the grid.
<i>xStep</i>	The horizontal spacing between grid cells.
<i>xNumCells</i>	The number of grid cell columns.
<i>yStart</i>	The minimum Y coordinate of the grid.
<i>yStep</i>	The vertical spacing between grid cells.
<i>yNumCells</i>	The number of grid cell rows.
<i>interp_table</i>	Pointer to an interpolation table. The table is created by the <code>mllib_ImageInterpTableCreate()</code> function.
<i>edge</i>	Type of edge condition. It can be one of the following: MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_SRC_PADDED

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageInterpTableCreate(3MLIB)`,
`mllib_ImageInterpTableDelete(3MLIB)`, `mllib_ImageGridWarp(3MLIB)`,
`mllib_ImageGridWarp_Fp(3MLIB)`, `mllib_ImageGridWarpTable(3MLIB)`,
`attributes(5)`

NAME	mllib_ImageHistogram2 – histogram
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageHistogram2(mllib_s32 **<i>histo</i>, const mllib_image *<i>img</i>, const mllib_s32 *<i>numBins</i>, const mllib_s32 *<i>lowValue</i>, const mllib_s32 *<i>highValue</i>, mllib_s32 <i>xStart</i>, mllib_s32 <i>yStart</i>, mllib_s32 <i>xPeriod</i>, mllib_s32 <i>yPeriod</i>);</pre>
DESCRIPTION	<p>The <code>mllib_ImageHistogram2()</code> function creates a histogram by scanning an image, counting the number of pixels within a given range for each channel of the image, and then generating a histogram.</p> <p>The image can have 1, 2, 3 or 4 channels. The data type of the image can be <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>. The histogram must have the same number of channels as the image has.</p> <p>One entry of the histogram, or a bin, is used to accumulate the number of pixels within a certain sub-range. The legal pixel range and the number of bins may be controlled separately.</p> <p>If <code>binWidth</code> is defined as $(highValue - lowValue) / numBins$ then bin <code>i</code> counts pixel values in the following range:</p> $lowValue + i * binWidth \leq x < lowValue + (i + 1) * binWidth$ <p>The set of pixels scanned may furthermore be reduced by specifying <code>xPeriod</code> and <code>yPeriod</code> parameters that specify the sampling rate along each axis.</p> <p>The set of pixels to be accumulated may be obtained from the following equation:</p> $x = xStart + p * xPeriod; \quad 0 \leq p < (w - xStart) / xPeriod$ $y = yStart + q * yPeriod; \quad 0 \leq q < (h - yStart) / yPeriod$ <p>It is the user's responsibility to clear the histogram table before this function is called and to ensure that the histogram table supplied is suitable for the source image and the parameters. Otherwise, the result of this function is undefined.</p> <p>The range from <code>lowValue[k]</code> to <code>(highValue[k] - 1)</code> must be a valid subrange of the image type range.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>histo</i> Pointer to histogram. The format of the histogram is <code>histo[channel][index]</code>. The index values for channel <code>i</code> can be <code>0, 1, ..., numBins[i]-1</code>.</p> <p><i>img</i> Pointer to source image.</p> <p><i>numBins</i> The number of bins for each channel of the image.</p> <p><i>lowValue</i> The lowest pixel value checked for each channel.</p>

mllib_ImageHistogram2(3MLIB)

<i>highValue</i>	The highest pixel value checked for each channel. When counting the pixel values, <i>highValue</i> is not included.
<i>xStart</i>	The initial X sample coordinate.
<i>yStart</i>	The initial Y sample coordinate.
<i>xPeriod</i>	The X sampling rate. $xPeriod \geq 1$.
<i>yPeriod</i>	The Y sampling rate. $yPeriod \geq 1$.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_ImageHistogram\(3MLIB\)](#), [attributes\(5\)](#)

NAME	mllib_ImageHistogram – histogram						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageHistogram(mllib_s32 <i>**histo</i>, const mllib_image <i>*img</i>);</pre>						
DESCRIPTION	The <code>mllib_ImageHistogram()</code> function creates a histogram. The data type of the image can be <code>MLIB_BYTE</code> , <code>MLIB_SHORT</code> , <code>MLIB_USHORT</code> , or <code>MLIB_INT</code> .						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>histo</i> Pointer to histogram. The format of the histogram is <code>histo[channel][index]</code>. The <code>MLIB_BYTE</code> type entries are indexed from 0 to 255. The <code>MLIB_SHORT</code> type entries are indexed from -32768 to -1, then from 0 to 32767. The <code>MLIB_USHORT</code> type entries are indexed from 0 to 65535. The <code>MLIB_INT</code> type entries are indexed from -2147483648 to -1, then from 0 to 2147483647.</p> <p><i>img</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageHistogram2(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageInterpTableCreate(3MLIB)

NAME	mllib_ImageInterpTableCreate – creates an interpolation table																				
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> void *mllib_ImageInterpTableCreate(mllib_type <i>type</i>, mllib_s32 <i>width</i>, mllib_s32 <i>height</i>, mllib_s32 <i>leftPadding</i>, mllib_s32 <i>topPadding</i>, mllib_s32 <i>subsampleBitsH</i>, mllib_s32 <i>subsampleBitsV</i>, mllib_s32 <i>precisionBits</i>, const void *<i>dataH</i>, const void *<i>dataV</i>);</pre>																				
DESCRIPTION	<p>The mllib_ImageInterpTableCreate() function creates an interpolation table based on parameters specified.</p> <p>This function creates an internal data structure, an interpolation table, which can be used by some image geometric functions for implementing a table-driven interpolation algorithm.</p> <p>The parameter <i>type</i> defines the type of <i>dataH</i>/<i>dataV</i> input arrays and can be MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, MLIB_INT, MLIB_FLOAT, or MLIB_DOUBLE.</p> <p>The <i>dataH</i> array should have at least $width * 2^{**subsampleBitsH}$ entries. <i>dataH</i>[$i * 2^{**subsampleBitsH}$] holds the coefficient for the leftmost neighboring pixel, <i>dataH</i>[$i * 2^{**subsampleBitsH} + 1$] holds the coefficient for the second neighboring pixel from left, ..., and <i>dataH</i>[$i * 2^{**subsampleBitsH} + width - 1$] holds the coefficient for the rightmost neighboring pixel, where $i = 0, 1, 2, \dots, 2^{**subsampleBitsH} - 1$.</p> <p>The <i>dataV</i> array should have at least $height * 2^{**subsampleBitsV}$ entries or should be NULL. If <i>dataV</i> is NULL, then <i>dataH</i> is used in its place, and in this case the parameters <i>topPadding</i>, <i>height</i>, and <i>subsampleBitsV</i> are ignored.</p>																				
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>type</i></td><td>Data type of the coefficients.</td></tr><tr><td><i>width</i></td><td>Width of the interpolation kernel in pixels.</td></tr><tr><td><i>height</i></td><td>Height of the interpolation kernel in pixels.</td></tr><tr><td><i>leftPadding</i></td><td>Number of pixels lying to the left of the interpolation kernel key position.</td></tr><tr><td><i>topPadding</i></td><td>Number of pixels lying above the interpolation kernel key position.</td></tr><tr><td><i>subsampleBitsH</i></td><td>Numbers of bits used for the horizontal subsample position.</td></tr><tr><td><i>subsampleBitsV</i></td><td>Numbers of bits used for the vertical subsample position.</td></tr><tr><td><i>precisionBits</i></td><td>Number of fractional bits used to describe the coefficients.</td></tr><tr><td><i>dataH</i></td><td>Pointer to horizontal coefficient data.</td></tr><tr><td><i>dataV</i></td><td>Pointer to vertical coefficient data.</td></tr></table>	<i>type</i>	Data type of the coefficients.	<i>width</i>	Width of the interpolation kernel in pixels.	<i>height</i>	Height of the interpolation kernel in pixels.	<i>leftPadding</i>	Number of pixels lying to the left of the interpolation kernel key position.	<i>topPadding</i>	Number of pixels lying above the interpolation kernel key position.	<i>subsampleBitsH</i>	Numbers of bits used for the horizontal subsample position.	<i>subsampleBitsV</i>	Numbers of bits used for the vertical subsample position.	<i>precisionBits</i>	Number of fractional bits used to describe the coefficients.	<i>dataH</i>	Pointer to horizontal coefficient data.	<i>dataV</i>	Pointer to vertical coefficient data.
<i>type</i>	Data type of the coefficients.																				
<i>width</i>	Width of the interpolation kernel in pixels.																				
<i>height</i>	Height of the interpolation kernel in pixels.																				
<i>leftPadding</i>	Number of pixels lying to the left of the interpolation kernel key position.																				
<i>topPadding</i>	Number of pixels lying above the interpolation kernel key position.																				
<i>subsampleBitsH</i>	Numbers of bits used for the horizontal subsample position.																				
<i>subsampleBitsV</i>	Numbers of bits used for the vertical subsample position.																				
<i>precisionBits</i>	Number of fractional bits used to describe the coefficients.																				
<i>dataH</i>	Pointer to horizontal coefficient data.																				
<i>dataV</i>	Pointer to vertical coefficient data.																				

`mllib_ImageInterpTableCreate(3MLIB)`

RETURN VALUES The function returns a pointer to an interpolation table.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageInterpTableDelete(3MLIB)`, `mllib_ImageAffineTable(3MLIB)`,
`mllib_ImageZoomTranslateTable(3MLIB)`,
`mllib_ImageGridWarpTable(3MLIB)`,
`mllib_ImagePolynomialWarpTable(3MLIB)`, `attributes(5)`

mllib_ImageInterpTableDelete(3MLIB)

NAME | mllib_ImageInterpTableDelete – deletes an interpolation table

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
void mllib_ImageInterpTableDelete(void *interp_table);
```

DESCRIPTION | The mllib_ImageInterpTableDelete() function deletes an interpolation table. This function deletes the structure of an interpolation table and frees the memory allocated by mllib_ImageInterpTableCreate().

PARAMETERS | The function takes the following arguments:
interp_table Pointer to an interpolation table.

RETURN VALUES | None.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_ImageInterpTableCreate(3MLIB), mllib_ImageAffineTable(3MLIB), mllib_ImageZoomTranslateTable(3MLIB), mllib_ImageGridWarpTable(3MLIB), mllib_ImagePolynomialWarpTable(3MLIB), attributes(5)

NAME	mllib_ImageInvert – invert																				
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageInvert(mllib_image *dst, const mllib_image *src);</pre>																				
DESCRIPTION	<p>The <code>mllib_ImageInvert()</code> function performs the inversion of an image such that white becomes black, light gray becomes dark gray, and so on.</p> <p>It uses the following equation:</p> $dst[x][y][i] = (Gwhite + Gblack) - src[x][y][i]$ <p>The values of <code>Gwhite</code> and <code>Gblack</code> for different types of images are:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Image Type</th> <th>Gwhite</th> <th>Gblack</th> <th>Gwhite + Gblack</th> </tr> </thead> <tbody> <tr> <td>MLIB_BYTE</td> <td>255</td> <td>0</td> <td>255 (0xFF)</td> </tr> <tr> <td>MLIB_SHORT</td> <td>32767</td> <td>-32768</td> <td>-1 (0xFFFF)</td> </tr> <tr> <td>MLIB_USHORT</td> <td>65535</td> <td>0</td> <td>65535 (0xFFFF)</td> </tr> <tr> <td>MLIB_INT</td> <td>2147483647</td> <td>-2147483648</td> <td>-1 (0xFFFFFFFF)</td> </tr> </tbody> </table> <p>Given that integer data are in the two's complement representation, <code>mllib_ImageInvert()</code> is the same as <code>mllib_ImageNot()</code>, while <code>mllib_ImageInvert_Inp()</code> is the same as <code>mllib_ImageNot_Inp()</code>.</p>	Image Type	Gwhite	Gblack	Gwhite + Gblack	MLIB_BYTE	255	0	255 (0xFF)	MLIB_SHORT	32767	-32768	-1 (0xFFFF)	MLIB_USHORT	65535	0	65535 (0xFFFF)	MLIB_INT	2147483647	-2147483648	-1 (0xFFFFFFFF)
Image Type	Gwhite	Gblack	Gwhite + Gblack																		
MLIB_BYTE	255	0	255 (0xFF)																		
MLIB_SHORT	32767	-32768	-1 (0xFFFF)																		
MLIB_USHORT	65535	0	65535 (0xFFFF)																		
MLIB_INT	2147483647	-2147483648	-1 (0xFFFFFFFF)																		
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>																				
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .																				
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:																				
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe														
ATTRIBUTE TYPE	ATTRIBUTE VALUE																				
Interface Stability	Evolving																				
MT-Level	MT-Safe																				
SEE ALSO	<code>mllib_ImageInvert_Inp(3MLIB)</code> , <code>mllib_ImageInvert_Fp(3MLIB)</code> , <code>mllib_ImageInvert_Fp_Inp(3MLIB)</code> , <code>attributes(5)</code>																				

mllib_ImageInvert_Fp(3MLIB)

NAME | mllib_ImageInvert_Fp – invert

SYNOPSIS | cc [*flag...*] *file...* -lmllib [*library...*]
| #include <mllib.h>

```
mllib_status mllib_ImageInvert_Fp(mllib_image *dst, const mllib_image  
| *src);
```

DESCRIPTION | The `mllib_ImageInvert_Fp()` function performs the floating-point inversion of an image such that white becomes black, light gray becomes dark gray, and so on.

| It uses the following equation:

```
dst[x][y][i] = -src[x][y][i]
```

PARAMETERS | The function takes the following arguments:

| *dst* | Pointer to destination image.

| *src* | Pointer to source image.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageInvert_Fp_Inp(3MLIB)`, `mllib_ImageInvert(3MLIB)`,
| `mllib_ImageInvert_Inp(3MLIB)`, `attributes(5)`

NAME mllib_ImageInvert_Fp_Inp – invert

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageInvert_Fp_Inp(mllib_image *srcdst);
```

DESCRIPTION The mllib_ImageInvert_Fp_Inp() function performs the floating-point inversion of an image such that white becomes black, light gray becomes dark gray, and so on, in place.

It uses the following equation:

$$srcdst[x][y][i] = -srcdst[x][y][i]$$

PARAMETERS The function takes the following arguments:

srcdst Pointer to source and destination image.

RETURN VALUES The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_ImageInvert_Fp\(3MLIB\)](#), [mllib_ImageInvert\(3MLIB\)](#), [mllib_ImageInvert_Inp\(3MLIB\)](#), [attributes\(5\)](#)

mllib_ImageInvert_Inp(3MLIB)

NAME	mllib_ImageInvert_Inp – invert in place																				
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageInvert_Inp(mllib_image *srcdst);</pre>																				
DESCRIPTION	<p>The <code>mllib_ImageInvert_Inp()</code> function performs the in-place inversion of an image such that white becomes black, light gray becomes dark gray, and so on.</p> <p>It uses the following equation:</p> $\text{srcdst}[x][y][i] = (\text{Gwhite} + \text{Gblack}) - \text{srcdst}[x][y][i]$ <p>The values of <code>Gwhite</code> and <code>Gblack</code> for different types of images are:</p> <table border="1"><thead><tr><th>Image Type</th><th>Gwhite</th><th>Gblack</th><th>Gwhite + Gblack</th></tr></thead><tbody><tr><td>MLIB_BYTE</td><td>255</td><td>0</td><td>255 (0xFF)</td></tr><tr><td>MLIB_SHORT</td><td>32767</td><td>-32768</td><td>-1 (0xFFFF)</td></tr><tr><td>MLIB_USHORT</td><td>65535</td><td>0</td><td>65535 (0xFFFF)</td></tr><tr><td>MLIB_INT</td><td>2147483647</td><td>-2147483648</td><td>-1 (0xFFFFFFFF)</td></tr></tbody></table> <p>Given that integer data are in the two's complement representation, <code>mllib_ImageInvert()</code> is the same as <code>mllib_ImageNot()</code>, while <code>mllib_ImageInvert_Inp()</code> is the same as <code>mllib_ImageNot_Inp()</code>.</p>	Image Type	Gwhite	Gblack	Gwhite + Gblack	MLIB_BYTE	255	0	255 (0xFF)	MLIB_SHORT	32767	-32768	-1 (0xFFFF)	MLIB_USHORT	65535	0	65535 (0xFFFF)	MLIB_INT	2147483647	-2147483648	-1 (0xFFFFFFFF)
Image Type	Gwhite	Gblack	Gwhite + Gblack																		
MLIB_BYTE	255	0	255 (0xFF)																		
MLIB_SHORT	32767	-32768	-1 (0xFFFF)																		
MLIB_USHORT	65535	0	65535 (0xFFFF)																		
MLIB_INT	2147483647	-2147483648	-1 (0xFFFFFFFF)																		
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>srcdst</i> Pointer to source and destination image.</p>																				
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .																				
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:																				
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe														
ATTRIBUTE TYPE	ATTRIBUTE VALUE																				
Interface Stability	Evolving																				
MT-Level	MT-Safe																				
SEE ALSO	<code>mllib_ImageInvert(3MLIB)</code> , <code>mllib_ImageInvert_Fp(3MLIB)</code> , <code>mllib_ImageInvert_Fp_Inp(3MLIB)</code> , <code>attributes(5)</code>																				

mlib_ImageIsNotAligned2(3MLIB)

NAME | mlib_ImageIsNotAligned2 – image query, two-byte aligned

SYNOPSIS | `cc [flag...] file... -lmlib [library...]`
`#include <mlib.h>`

`int mlib_ImageIsNotAligned2(const mlib_image *img);`

DESCRIPTION | The `mlib_ImageIsNotAligned2()` function tests for a specific alignment of a mediaLib image structure.

PARAMETERS | The function takes the following arguments:
img Pointer to source image.

RETURN VALUES | Returns 0 if data address is two-byte aligned; otherwise, returns nonzero.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mlib_ImageIsNotAligned4(3MLIB)`, `mlib_ImageIsNotAligned8(3MLIB)`,
`mlib_ImageIsNotAligned64(3MLIB)`, `mlib_ImageIsNotHeight2X(3MLIB)`,
`mlib_ImageIsNotHeight4X(3MLIB)`, `mlib_ImageIsNotHeight8X(3MLIB)`,
`mlib_ImageIsNotOneDvector(3MLIB)`, `mlib_ImageIsNotStride8X(3MLIB)`,
`mlib_ImageIsNotWidth2X(3MLIB)`, `mlib_ImageIsNotWidth4X(3MLIB)`,
`mlib_ImageIsNotWidth8X(3MLIB)`, `mlib_ImageIsUserAllocated(3MLIB)`,
`attributes(5)`

mllib_ImageIsNotAligned4(3MLIB)

NAME mllib_ImageIsNotAligned4 – image query, four-byte aligned

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

int mllib_ImageIsNotAligned4 (const mllib_image *img);
```

DESCRIPTION The mllib_ImageIsNotAligned4 () function tests for a specific alignment of a mediaLib image structure.

PARAMETERS The function takes the following arguments:

img Pointer to source image.

RETURN VALUES Returns 0 if data address is four-byte aligned; otherwise, returns nonzero.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageIsNotAligned2(3MLIB), mllib_ImageIsNotAligned8(3MLIB), mllib_ImageIsNotAligned64(3MLIB), mllib_ImageIsNotHeight2X(3MLIB), mllib_ImageIsNotHeight4X(3MLIB), mllib_ImageIsNotHeight8X(3MLIB), mllib_ImageIsNotOneDvector(3MLIB), mllib_ImageIsNotStride8X(3MLIB), mllib_ImageIsNotWidth2X(3MLIB), mllib_ImageIsNotWidth4X(3MLIB), mllib_ImageIsNotWidth8X(3MLIB), mllib_ImageIsUserAllocated(3MLIB), attributes(5)

mllib_ImageIsNotAligned64(3MLIB)

NAME mllib_ImageIsNotAligned64 – image query, 64-byte aligned

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

int mllib_ImageIsNotAligned64 (const mllib_image *img);
```

DESCRIPTION The mllib_ImageIsNotAligned64 () function tests for a specific alignment characteristic of a mediaLib image structure.

PARAMETERS The function takes the following arguments:
img Pointer to source image.

RETURN VALUES Returns 0 if data address is 64-byte aligned; otherwise, returns nonzero.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageIsNotAligned2(3MLIB), mllib_ImageIsNotAligned4(3MLIB), mllib_ImageIsNotAligned8(3MLIB), mllib_ImageIsNotHeight2X(3MLIB), mllib_ImageIsNotHeight4X(3MLIB), mllib_ImageIsNotHeight8X(3MLIB), mllib_ImageIsNotOneDvector(3MLIB), mllib_ImageIsNotStride8X(3MLIB), mllib_ImageIsNotWidth2X(3MLIB), mllib_ImageIsNotWidth4X(3MLIB), mllib_ImageIsNotWidth8X(3MLIB), mllib_ImageIsUserAllocated(3MLIB), attributes(5)

mllib_ImageIsNotAligned8(3MLIB)

NAME mllib_ImageIsNotAligned8 – image query, eight-byte aligned

SYNOPSIS cc [*flag...*] *file...* -lmllib [*library...*]
#include <mllib.h>

```
int mllib_ImageIsNotAligned8 (const mllib_image *img);
```

DESCRIPTION The mllib_ImageIsNotAligned8 () function tests for a specific alignment of a mediaLib image structure.

PARAMETERS The function takes the following arguments:

img Pointer to source image.

RETURN VALUES Returns 0 if data address is eight-byte aligned; otherwise, returns nonzero.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageIsNotAligned2(3MLIB), mllib_ImageIsNotAligned4(3MLIB), mllib_ImageIsNotAligned64(3MLIB), mllib_ImageIsNotHeight2X(3MLIB), mllib_ImageIsNotHeight4X(3MLIB), mllib_ImageIsNotHeight8X(3MLIB), mllib_ImageIsNotOneDvector(3MLIB), mllib_ImageIsNotStride8X(3MLIB), mllib_ImageIsNotWidth2X(3MLIB), mllib_ImageIsNotWidth4X(3MLIB), mllib_ImageIsNotWidth8X(3MLIB), mllib_ImageIsUserAllocated(3MLIB), attributes(5)

mlib_ImageIsNotHeight2X(3MLIB)

NAME | mlib_ImageIsNotHeight2X – image query, 2X height

SYNOPSIS | `cc [flag...] file... -lmlib [library...]`
`#include <mlib.h>`

`int mlib_ImageIsNotHeight2X(const mlib_image *img);`

DESCRIPTION | The `mlib_ImageIsNotHeight2X()` function tests for a specific height characteristic of a `mediaLib` image structure.

PARAMETERS | The function takes the following arguments:
img Pointer to source image.

RETURN VALUES | Returns 0 if height is a multiple of two; otherwise, returns nonzero.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mlib_ImageIsNotAligned2(3MLIB)`, `mlib_ImageIsNotAligned4(3MLIB)`,
`mlib_ImageIsNotAligned8(3MLIB)`, `mlib_ImageIsNotAligned64(3MLIB)`,
`mlib_ImageIsNotHeight4X(3MLIB)`, `mlib_ImageIsNotHeight8X(3MLIB)`,
`mlib_ImageIsNotOneDvector(3MLIB)`, `mlib_ImageIsNotStride8X(3MLIB)`,
`mlib_ImageIsNotWidth2X(3MLIB)`, `mlib_ImageIsNotWidth4X(3MLIB)`,
`mlib_ImageIsNotWidth8X(3MLIB)`, `mlib_ImageIsUserAllocated(3MLIB)`,
`attributes(5)`

mllib_ImageIsNotHeight4X(3MLIB)

NAME | mllib_ImageIsNotHeight4X – image query, 4X height

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
int mllib_ImageIsNotHeight4X(const mllib_image *img);
```

DESCRIPTION | The mllib_ImageIsNotHeight4X() function tests for a specific height characteristic of a mediaLib image structure.

PARAMETERS | The function takes the following arguments:
img Pointer to source image.

RETURN VALUES | Returns 0 if height is a multiple of four; otherwise, returns nonzero.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_ImageIsNotAligned2(3MLIB), mllib_ImageIsNotAligned4(3MLIB), mllib_ImageIsNotAligned8(3MLIB), mllib_ImageIsNotAligned64(3MLIB), mllib_ImageIsNotHeight2X(3MLIB), mllib_ImageIsNotHeight8X(3MLIB), mllib_ImageIsNotOneDvector(3MLIB), mllib_ImageIsNotStride8X(3MLIB), mllib_ImageIsNotWidth2X(3MLIB), mllib_ImageIsNotWidth4X(3MLIB), mllib_ImageIsNotWidth8X(3MLIB), mllib_ImageIsUserAllocated(3MLIB), attributes(5)

mlib_ImageIsNotHeight8X(3MLIB)

NAME | mlib_ImageIsNotHeight8X – image query, 8X height

SYNOPSIS | `cc [flag...] file... -lmlib [library...]`
`#include <mlib.h>`

`int mlib_ImageIsNotHeight8X(const mlib_image *img);`

DESCRIPTION | The `mlib_ImageIsNotHeight8X()` function tests for a specific height characteristic of a `mediaLib` image structure.

PARAMETERS | The function takes the following arguments:
img Pointer to source image.

RETURN VALUES | Returns 0 if height is a multiple of eight; otherwise, returns nonzero.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mlib_ImageIsNotAligned2(3MLIB)`, `mlib_ImageIsNotAligned4(3MLIB)`,
`mlib_ImageIsNotAligned8(3MLIB)`, `mlib_ImageIsNotAligned64(3MLIB)`,
`mlib_ImageIsNotHeight2X(3MLIB)`, `mlib_ImageIsNotHeight4X(3MLIB)`,
`mlib_ImageIsNotOneDvector(3MLIB)`, `mlib_ImageIsNotStride8X(3MLIB)`,
`mlib_ImageIsNotWidth2X(3MLIB)`, `mlib_ImageIsNotWidth4X(3MLIB)`,
`mlib_ImageIsNotWidth8X(3MLIB)`, `mlib_ImageIsUserAllocated(3MLIB)`,
`attributes(5)`

mllib_ImageIsNotOneDvector(3MLIB)

NAME mllib_ImageIsNotOneDvector – image query, 1D vector

SYNOPSIS cc [*flag...*] *file...* -lmllib [*library...*]
#include <mllib.h>

```
int mllib_ImageIsNotOneDvector(const mllib_image *img);
```

DESCRIPTION The mllib_ImageIsNotOneDvector() function tests for a specific dimension characteristic of a mediaLib image structure.

PARAMETERS The function takes the following arguments:

img Pointer to source image.

RETURN VALUES Returns 0 if data space can be treated as a 1D vector; otherwise, returns nonzero.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageIsNotAligned2(3MLIB), mllib_ImageIsNotAligned4(3MLIB), mllib_ImageIsNotAligned8(3MLIB), mllib_ImageIsNotAligned64(3MLIB), mllib_ImageIsNotHeight2X(3MLIB), mllib_ImageIsNotHeight4X(3MLIB), mllib_ImageIsNotHeight8X(3MLIB), mllib_ImageIsNotStride8X(3MLIB), mllib_ImageIsNotWidth2X(3MLIB), mllib_ImageIsNotWidth4X(3MLIB), mllib_ImageIsNotWidth8X(3MLIB), mllib_ImageIsUserAllocated(3MLIB), attributes(5)

mllib_ImageIsNotStride8X(3MLIB)

NAME mllib_ImageIsNotStride8X – image query, 8X stride

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
int mllib_ImageIsNotStride8X(const mllib_image *img);
```

DESCRIPTION The mllib_ImageIsNotStride8X() function tests for a specific stride characteristic of a mediaLib image structure.

PARAMETERS The function takes the following arguments:
img Pointer to source image.

RETURN VALUES Returns 0 if stride is a multiple of eight; otherwise, returns nonzero.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageIsNotAligned2(3MLIB), mllib_ImageIsNotAligned4(3MLIB), mllib_ImageIsNotAligned8(3MLIB), mllib_ImageIsNotAligned64(3MLIB), mllib_ImageIsNotHeight2X(3MLIB), mllib_ImageIsNotHeight4X(3MLIB), mllib_ImageIsNotHeight8X(3MLIB), mllib_ImageIsNotOneDvector(3MLIB), mllib_ImageIsNotWidth2X(3MLIB), mllib_ImageIsNotWidth4X(3MLIB), mllib_ImageIsNotWidth8X(3MLIB), mllib_ImageIsUserAllocated(3MLIB), attributes(5)

mllib_ImageIsNotWidth2X(3MLIB)

NAME mllib_ImageIsNotWidth2X – image query, 2X width

SYNOPSIS cc [*flag...*] *file...* -lmllib [*library...*]
#include <mllib.h>

```
int mllib_ImageIsNotWidth2X(const mllib_image *img);
```

DESCRIPTION The mllib_ImageIsNotWidth2X() function tests for a specific width characteristic of a mediaLib image structure.

PARAMETERS The function takes the following arguments:

img Pointer to source image.

RETURN VALUES Returns 0 if width is a multiple of two; otherwise, returns nonzero.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageIsNotAligned2(3MLIB), mllib_ImageIsNotAligned4(3MLIB), mllib_ImageIsNotAligned8(3MLIB), mllib_ImageIsNotAligned64(3MLIB), mllib_ImageIsNotHeight2X(3MLIB), mllib_ImageIsNotHeight4X(3MLIB), mllib_ImageIsNotHeight8X(3MLIB), mllib_ImageIsNotOneDvector(3MLIB), mllib_ImageIsNotStride8X(3MLIB), mllib_ImageIsNotWidth4X(3MLIB), mllib_ImageIsNotWidth8X(3MLIB), mllib_ImageIsUserAllocated(3MLIB), attributes(5)

mllib_ImageIsNotWidth4X(3MLIB)

NAME mllib_ImageIsNotWidth4X – image query, 4X width

SYNOPSIS `cc [flag...] file... -lmllib [library...]`
`#include <mllib.h>`

`int mllib_ImageIsNotWidth4X(const mllib_image *img);`

DESCRIPTION The mllib_ImageIsNotWidth4X() function tests for a specific width characteristic of a mediaLib image structure.

PARAMETERS The function takes the following arguments:

img Pointer to source image.

RETURN VALUES Returns 0 if width is a multiple of four; otherwise, returns nonzero.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageIsNotAligned2(3MLIB), mllib_ImageIsNotAligned4(3MLIB), mllib_ImageIsNotAligned8(3MLIB), mllib_ImageIsNotAligned64(3MLIB), mllib_ImageIsNotHeight2X(3MLIB), mllib_ImageIsNotHeight4X(3MLIB), mllib_ImageIsNotHeight8X(3MLIB), mllib_ImageIsNotOneDvector(3MLIB), mllib_ImageIsNotStride8X(3MLIB), mllib_ImageIsNotWidth2X(3MLIB), mllib_ImageIsNotWidth8X(3MLIB), mllib_ImageIsUserAllocated(3MLIB), attributes(5)

mllib_ImageIsNotWidth8X(3MLIB)

NAME mllib_ImageIsNotWidth8X – image query, 8X width

SYNOPSIS cc [*flag...*] *file...* -lmllib [*library...*]
#include <mllib.h>

```
int mllib_ImageIsNotWidth8X(const mllib_image *img);
```

DESCRIPTION The mllib_ImageIsNotWidth8X() function tests for a specific width characteristic of a mediaLib image structure.

PARAMETERS The function takes the following arguments:

img Pointer to source image.

RETURN VALUES Returns 0 if width is a multiple of eight; otherwise, returns nonzero.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageIsNotAligned2(3MLIB), mllib_ImageIsNotAligned4(3MLIB), mllib_ImageIsNotAligned8(3MLIB), mllib_ImageIsNotAligned64(3MLIB), mllib_ImageIsNotHeight2X(3MLIB), mllib_ImageIsNotHeight4X(3MLIB), mllib_ImageIsNotHeight8X(3MLIB), mllib_ImageIsNotOneDvector(3MLIB), mllib_ImageIsNotStride8X(3MLIB), mllib_ImageIsNotWidth2X(3MLIB), mllib_ImageIsNotWidth4X(3MLIB), mllib_ImageIsUserAllocated(3MLIB), attributes(5)

mllib_ImageIsUserAllocated(3MLIB)

NAME mllib_ImageIsUserAllocated – image query, user-allocated

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
int mllib_ImageIsUserAllocated(const mllib_image *src);
```

DESCRIPTION The mllib_ImageIsUserAllocated() function tests for a specific allocation characteristic of a mediaLib image structure.

PARAMETERS The function takes the following arguments:
src Pointer to source image.

RETURN VALUES Returns 0 if data space has been allocated by mediaLib; otherwise, returns nonzero.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageIsNotAligned2(3MLIB), mllib_ImageIsNotAligned4(3MLIB), mllib_ImageIsNotAligned8(3MLIB), mllib_ImageIsNotAligned64(3MLIB), mllib_ImageIsNotHeight2X(3MLIB), mllib_ImageIsNotHeight4X(3MLIB), mllib_ImageIsNotHeight8X(3MLIB), mllib_ImageIsNotOneDvector(3MLIB), mllib_ImageIsNotStride8X(3MLIB), mllib_ImageIsNotWidth2X(3MLIB), mllib_ImageIsNotWidth4X(3MLIB), mllib_ImageIsNotWidth8X(3MLIB), attributes(5)

mllib_ImageLog(3MLIB)

NAME | mllib_ImageLog – computes the natural logarithm of the image pixels

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
mllib_status mllib_ImageLog(mllib_image *dst, const mllib_image *src);
```

DESCRIPTION | The mllib_ImageLog() function computes the natural logarithm of the image pixels.
It uses the following equation:
$$dst[x][y][i] = \log(src[x][y][i])$$

PARAMETERS | The function takes the following arguments:
dst | Pointer to destination image.
src | Pointer to source image.

RETURN VALUES | The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_ImageLog_Fp\(3MLIB\)](#), [mllib_ImageLog_Fp_Inp\(3MLIB\)](#), [mllib_ImageLog_Inp\(3MLIB\)](#), [attributes\(5\)](#)

NAME mllib_ImageLog_Fp – computes the natural logarithm of the image pixels

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageLog_Fp(mllib_image *dst, const mllib_image
    *src);
```

DESCRIPTION The mllib_ImageLog_Fp() function computes the natural logarithm of the image pixels.

It uses the following equation:

$$dst[x][y][i] = \log(src[x][y][i])$$

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_ImageLog\(3MLIB\)](#), [mllib_ImageLog_Fp_Inp\(3MLIB\)](#), [mllib_ImageLog_Inp\(3MLIB\)](#), [attributes\(5\)](#)

mllib_ImageLog_Fp_Inp(3MLIB)

NAME	mllib_ImageLog_Fp_Inp – computes the natural logarithm of the image pixels
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageLog_Fp_Inp(mllib_image *srcdst);</pre>
DESCRIPTION	<p>The <code>mllib_ImageLog_Fp_Inp()</code> function computes the natural logarithm of the image pixels, in place.</p> <p>It uses the following equation:</p> $\text{srcdst}[x][y][i] = \log(\text{srcdst}[x][y][i])$
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>srcdst</i> Pointer to source and destination image.</p>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_ImageLog\(3MLIB\)](#), [mllib_ImageLog_Fp\(3MLIB\)](#), [mllib_ImageLog_Inp\(3MLIB\)](#), [attributes\(5\)](#)

NAME | mllib_ImageLog_Inp – computes the natural logarithm of the image pixels, in place

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageLog_Inp(mllib_image *srcdst);
```

DESCRIPTION | The mllib_ImageLog_Inp() function computes the natural logarithm of the image pixels in place.

It uses the following equation:

$$\text{srcdst}[x][y][i] = \log(\text{srcdst}[x][y][i])$$

PARAMETERS | The function takes the following arguments:

srcdst Pointer to source and destination image.

RETURN VALUES | The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_ImageLog\(3MLIB\)](#), [mllib_ImageLog_Fp\(3MLIB\)](#), [mllib_ImageLog_Fp_Inp\(3MLIB\)](#), [attributes\(5\)](#)

mllib_ImageLookup2(3MLIB)

NAME | mllib_ImageLookup2 – table lookup

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageLookup2(mllib_image *dst, const mllib_image
    *src, const void **table, const mllib_s32 *offsets, mllib_s32
    channels);
```

DESCRIPTION | The `mllib_ImageLookup2()` function maps the source image to the destination image by using the user-specified lookup table and an offset.

The source and destination images must have the same width and height.

The source and destination images can have different data types. See the following table for available variations of the table lookup function on image types:

Type [*]	BYTE	SHORT	USHORT	INT	FLOAT	DOUBLE
MLIB_BIT	Y					
MLIB_BYTE	Y	Y	Y	Y	Y	Y
MLIB_SHORT	Y	Y	Y	Y	Y	Y
MLIB_USHORT	Y	Y	Y	Y	Y	Y
MLIB_INT	Y	Y	Y	Y	Y	Y

[*] Each row represents a source data type. Each column represents a destination data type.

The source and destination images also can have a different number of channels. The source image can be a single-channel image or can have the same number of channels as the destination image. The lookup table can have one channel or have the same channels as the destination image. See the following table for possible variations on the number of channels in the images and the lookup table:

# of channels in the input image	# of channels in the lookup table	# of channels in the output image
1	n	n
n	1	n
n	n	n

where, $n = 1, 2, 3, 4$.

Each of the following equations is used in the corresponding case shown in the table above.

```
dst[x][y][i] = table[i][src[x][y][0] - offsets[i]]
dst[x][y][i] = table[0][src[x][y][i] - offsets[0]]
dst[x][y][i] = table[i][src[x][y][i] - offsets[i]]
```

PARAMETERS

The function takes the following arguments:

- dst* Pointer to destination image.
- src* Pointer to source image.
- table* Pointer to lookup table. The data type of the lookup table is the same as that of the destination image.. The format of the lookup table is:

 $table[channel][index]$

 The entries are indexed from 0 to 1, 2, ..., and so on. It is the user's responsibility to provide a lookup table that has enough entries to cover all possible values of the pixel components deducted by the offset in each channel of the source image.
- offsets* Offset values subtracted from the src pixel before table lookup.
- channels* Number of channels in the lookup table. If the number of channels equals 1, then the same table is applied to all channels. Otherwise, the number of channels must be no less than the number of channels in the destination image.

RETURN VALUES

The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

`mllib_ImageLookUp(3MLIB)`, `mllib_ImageLookUp_Inp(3MLIB)`, `mllib_ImageLookUpMask(3MLIB)`, `attributes(5)`

mllib_ImageLookup(3MLIB)

NAME	mllib_ImageLookup – table lookup																																										
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageLookup(mllib_image *dst, const mllib_image *src, const void **table);</pre>																																										
DESCRIPTION	<p>The <code>mllib_ImageLookup()</code> function maps the source image to the destination image by using the user-specified lookup table.</p> <p>The source and destination images must have the same width and height. The source image can be a single channel image or can have the same number of channels as the destination image. One of the following equations is used accordingly:</p> <pre>dst[x][y][i] = table[i][src[x][y][0]] dst[x][y][i] = table[i][src[x][y][i]]</pre> <p>The source and destination images can have different data types. See the following table for available variations of the table lookup function on image types:</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th>Type [*]</th> <th>BYTE</th> <th>SHORT</th> <th>USHORT</th> <th>INT</th> <th>FLOAT</th> <th>DOUBLE</th> </tr> </thead> <tbody> <tr> <td>MLIB_BIT</td> <td>Y</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>MLIB_BYTE</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> </tr> <tr> <td>MLIB_SHORT</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> </tr> <tr> <td>MLIB_USHORT</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> </tr> <tr> <td>MLIB_INT</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> </tr> </tbody> </table> <p>[*] Each row represents a source data type. Each column represents a destination data type.</p>	Type [*]	BYTE	SHORT	USHORT	INT	FLOAT	DOUBLE	MLIB_BIT	Y						MLIB_BYTE	Y	Y	Y	Y	Y	Y	MLIB_SHORT	Y	Y	Y	Y	Y	Y	MLIB_USHORT	Y	Y	Y	Y	Y	Y	MLIB_INT	Y	Y	Y	Y	Y	Y
Type [*]	BYTE	SHORT	USHORT	INT	FLOAT	DOUBLE																																					
MLIB_BIT	Y																																										
MLIB_BYTE	Y	Y	Y	Y	Y	Y																																					
MLIB_SHORT	Y	Y	Y	Y	Y	Y																																					
MLIB_USHORT	Y	Y	Y	Y	Y	Y																																					
MLIB_INT	Y	Y	Y	Y	Y	Y																																					
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>table</i> Pointer to lookup table. The data type of the lookup table is the same as the destination image. The number of entries in the lookup table is determined by the type of the input image. The format of the lookup table is:</p> <pre>table[channel][index]</pre> <p>The <code>MLIB_BIT</code> type entries are indexed from 0 to 1. The <code>MLIB_BYTE</code> type entries are indexed from 0 to 255. The <code>MLIB_SHORT</code> type entries are indexed from -32768 to -1, then from 0 to 32767. The <code>MLIB_USHORT</code> type entries are indexed from 0 to</p>																																										

`mllib_ImageLookUp(3MLIB)`

65535. The `MLIB_INT` type entries are indexed from -2147483648 to -1, and then from 0 to 2147483647.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageLookUp_Inp(3MLIB)`, `mllib_ImageLookUp2(3MLIB)`, `mllib_ImageLookUpMask(3MLIB)`, `attributes(5)`

mllib_ImageLookup_Inp(3MLIB)

NAME	mllib_ImageLookup_Inp – table lookup, in place						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageLookup_Inp(mllib_image *srcdst, const void **table) ;</pre>						
DESCRIPTION	<p>The <code>mllib_ImageLookup_Inp()</code> function maps the source image to the destination image, in place, by using the user-specified lookup table.</p> <p>The following equation is used:</p> $\text{srcdst}[x][y][i] = \text{table}[i][\text{srcdst}[x][y][i]]$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>srcdst</i> Pointer to first source and destination image.</p> <p><i>table</i> Pointer to lookup table. The data type of the lookup table is the same as the destination image. The number of entries in the lookup table is determined by the type of the input image. The format of the lookup table is:</p> $\text{table}[\text{channel}][\text{index}]$ <p>The <code>MLIB_BYTE</code> type entries are indexed from 0 to 255. The <code>MLIB_SHORT</code> type entries are indexed from -32768 to -1, then from 0 to 32767. The <code>MLIB_USHORT</code> type entries are indexed from 0 to 65535. The <code>MLIB_INT</code> type entries are indexed from -2147483648 to -1, and then from 0 to 2147483647.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageLookup(3MLIB)</code> , <code>mllib_ImageLookup2(3MLIB)</code> , <code>mllib_ImageLookupMask(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageLookUpMask – table lookup with mask

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageLookUpMask(mllib_image *dst, const
    mllib_image *src, const void **table, mllib_s32 channels, mllib_s32
    cmask);
```

DESCRIPTION The `mllib_ImageLookUpMask()` function maps the source image to the destination image by using the user-specified lookup table and applying a channel mask.

The source and destination images must have the same width and height. The source image can be a single channel image or can have the same number of channels as the destination image. One of the following equations is used accordingly:

$$dst[x][y][i] = table[i][src[x][y][0]]$$

$$dst[x][y][i] = table[i][src[x][y][i]]$$

The source and destination images can have different data types. See the following table for available variations of the table lookup function on image types:

Type [*]	BYTE	SHORT	USHORT	INT	FLOAT	DOUBLE
MLIB_BIT	Y					
MLIB_BYTE	Y	Y	Y	Y	Y	Y
MLIB_SHORT	Y	Y	Y	Y	Y	Y
MLIB_USHORT	Y	Y	Y	Y	Y	Y
MLIB_INT	Y	Y	Y	Y	Y	Y

[*] Each row represents a source data type. Each column represents a destination data type.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

table Pointer to lookup table. The data type of the lookup table is the same as the destination image. The number of entries in the lookup table is determined by the type of the input image. The format of the lookup table is:

```
table[channel][index]
```

The `MLIB_BIT` type entries are indexed from 0 to 1. The `MLIB_BYTE` type entries are indexed from 0 to 255. The `MLIB_SHORT` type entries are indexed from -32768 to -1, then from

mllib_ImageLookUpMask(3MLIB)

0 to 32767. The `MLIB_USHORT` type entries are indexed from 0 to 65535. The `MLIB_INT` type entries are indexed from -2147483648 to -1, and then from 0 to 2147483647.

channels Number of channels in the lookup table. If the number of channels is equal to 1, then the same table is applied to all channels. Otherwise, the number of channels must be no less than the number of valid 1s in the channel mask.

cmask Channel mask. Each bit of the mask represents a channel of an image or a lookup table. Only the rightmost four bits of *cmask* are considered, where the least significant bit of *cmask* is for the last channel. The channels corresponding to 0 bits of *cmask* are not processed or used. *cmask* is always applied to the destination image *dst*. If the source image *src* has the same number of channels as *dst*, then *cmask* is also applied to *src*. Otherwise, each channel of *src* is used for each *cmask* bit with a value of 1, in this order: the first channel for the first 1 from the left in *cmask*. If *src* has only one channel, then the same *src* channel is used for every *cmask* bit with a value of 1. If the lookup table has the same number of channels as *dst*, then *cmask* is also applied to *table*. Otherwise, each table channel is used for each *cmask* bit with a value of 1, in this order: the first channel for the first 1 from the left in *cmask*. If *table* has only one channel, then the same table channel is used for every *cmask* bit with a value of 1.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageLookUp(3MLIB)`, `mllib_ImageLookUp_Inp(3MLIB)`, `mllib_ImageLookUp2(3MLIB)`, `attributes(5)`

NAME mlib_ImageMax – two image maximum

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_ImageMax(mlib_image *dst, const mlib_image *src1,
    const mlib_image *src2);
```

DESCRIPTION The `mlib_ImageMax()` function accepts input from the two source images and writes the maximum to the destination image on a pixel-by-pixel basis.

It uses the following equation:

$$dst[x][y][i] = \text{MAX}\{ src1[x][y][i], src2[x][y][i] \}$$

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src1 Pointer to first source image.

src2 Pointer to second source image.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mlib_ImageMax_Fp(3MLIB)`, `mlib_ImageMax_Fp_Inp(3MLIB)`, `mlib_ImageMax_Inp(3MLIB)`, `attributes(5)`

mllib_ImageMaxFilter3x3(3MLIB)

NAME	mllib_ImageMaxFilter3x3 – 3x3 maximum filter
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageMaxFilter3x3(mllib_image *dst, const mllib_image *src);</pre>
DESCRIPTION	<p>The <code>mllib_ImageMaxFilter3x3()</code> function replaces the center pixel in a neighborhood with the maximum value in that neighborhood for each 3x3 neighborhood in the image.</p> <p>The source and destination images must be single-channel images.</p> <p>It uses the following equation:</p> $dst[x][y][0] = \text{MAX}\{ src[p][q][0], \quad x-1 \leq p \leq x+1; y-1 \leq q \leq y+1 \}$ <p>where $x = 1, \dots, w - 2$; $y = 1, \dots, h - 2$.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageMaxFilter3x3_Fp(3MLIB)`, `mllib_ImageMaxFilter5x5(3MLIB)`, `mllib_ImageMaxFilter5x5_Fp(3MLIB)`, `mllib_ImageMaxFilter7x7(3MLIB)`, `mllib_ImageMaxFilter7x7_Fp(3MLIB)`, `mllib_ImageMedianFilter3x3(3MLIB)`, `mllib_ImageMedianFilter3x3_Fp(3MLIB)`, `mllib_ImageMedianFilter3x3_US(3MLIB)`, `mllib_ImageMedianFilter5x5(3MLIB)`, `mllib_ImageMedianFilter5x5_Fp(3MLIB)`, `mllib_ImageMedianFilter5x5_US(3MLIB)`, `mllib_ImageMedianFilter7x7(3MLIB)`, `mllib_ImageMedianFilter7x7_Fp(3MLIB)`, `mllib_ImageMedianFilter7x7_US(3MLIB)`, `mllib_ImageMedianFilterMxN(3MLIB)`, `mllib_ImageMedianFilterMxN_Fp(3MLIB)`, `mllib_ImageMedianFilterMxN_US(3MLIB)`, `mllib_ImageMinFilter3x3(3MLIB)`, `mllib_ImageMinFilter3x3_Fp(3MLIB)`, `mllib_ImageMinFilter5x5(3MLIB)`,

`mllib_ImageMaxFilter3x3(3MLIB)`

```
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7(3MLIB),  
mllib_ImageMinFilter7x7_Fp(3MLIB), mllib_ImageRankFilter3x3(3MLIB),  
mllib_ImageRankFilter3x3_Fp(3MLIB),  
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),  
mllib_ImageRankFilter5x5_Fp(3MLIB),  
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),  
mllib_ImageRankFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),  
mllib_ImageRankFilterMxN_Fp(3MLIB),  
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

mllib_ImageMaxFilter3x3_Fp(3MLIB)

NAME	mllib_ImageMaxFilter3x3_Fp – 3x3 maximum filter						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageMaxFilter3x3_Fp(mllib_image *dst, const mllib_image *src);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageMaxFilter3x3_Fp()</code> function replaces the center pixel in a neighborhood with the floating-point maximum value in that neighborhood for each 3x3 neighborhood in the image.</p> <p>The source and destination images must be single-channel images.</p> <p>It uses the following equation:</p> $dst[x][y][0] = \text{MAX}\{ src[p][q][0], \quad x-1 \leq p \leq x+1; y-1 \leq q \leq y+1 \}$ <p>where $x = 1, \dots, w - 2$; $y = 1, \dots, h - 2$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<p><code>mllib_ImageMaxFilter3x3(3MLIB)</code>, <code>mllib_ImageMaxFilter5x5(3MLIB)</code>, <code>mllib_ImageMaxFilter5x5_Fp(3MLIB)</code>, <code>mllib_ImageMaxFilter7x7(3MLIB)</code>, <code>mllib_ImageMaxFilter7x7_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3_US(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5_US(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7_US(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN_US(3MLIB)</code>, <code>mllib_ImageMinFilter3x3(3MLIB)</code>, <code>mllib_ImageMinFilter3x3_Fp(3MLIB)</code>, <code>mllib_ImageMinFilter5x5(3MLIB)</code>,</p>						

mllib_ImageMaxFilter3x3_Fp(3MLIB)

```
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7(3MLIB),  
mllib_ImageMinFilter7x7_Fp(3MLIB), mllib_ImageRankFilter3x3(3MLIB),  
mllib_ImageRankFilter3x3_Fp(3MLIB),  
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),  
mllib_ImageRankFilter5x5_Fp(3MLIB),  
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),  
mllib_ImageRankFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),  
mllib_ImageRankFilterMxN_Fp(3MLIB),  
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

mllib_ImageMaxFilter5x5(3MLIB)

NAME	mllib_ImageMaxFilter5x5 – 5x5 Max Filter						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageMaxFilter5x5(mllib_image *dst, const mllib_image *src);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageMaxFilter5x5()</code> function replaces the center pixel in a neighborhood with the maximum value in that neighborhood for each 5x5 neighborhood in the image.</p> <p>The source and destination images must be single-channel images.</p> <p>It uses the following equation:</p> $dst[x][y][0] = \text{MAX}\{ src[p][q][0], \quad x-2 \leq p \leq x+2; y-2 \leq q \leq y+2 \}$ <p>where $x = 2, \dots, w - 3$; $y = 2, \dots, h - 3$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<p><code>mllib_ImageMaxFilter3x3(3MLIB)</code>, <code>mllib_ImageMaxFilter3x3_Fp(3MLIB)</code>, <code>mllib_ImageMaxFilter5x5_Fp(3MLIB)</code>, <code>mllib_ImageMaxFilter7x7(3MLIB)</code>, <code>mllib_ImageMaxFilter7x7_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3_US(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5_US(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7_US(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN_US(3MLIB)</code>, <code>mllib_ImageMinFilter3x3(3MLIB)</code>, <code>mllib_ImageMinFilter3x3_Fp(3MLIB)</code>, <code>mllib_ImageMinFilter5x5(3MLIB)</code>,</p>						

`mllib_ImageMaxFilter5x5(3MLIB)`

```
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7(3MLIB),  
mllib_ImageMinFilter7x7_Fp(3MLIB), mllib_ImageRankFilter3x3(3MLIB),  
mllib_ImageRankFilter3x3_Fp(3MLIB),  
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),  
mllib_ImageRankFilter5x5_Fp(3MLIB),  
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),  
mllib_ImageRankFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),  
mllib_ImageRankFilterMxN_Fp(3MLIB),  
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

mllib_ImageMaxFilter5x5_Fp(3MLIB)

NAME	mllib_ImageMaxFilter5x5_Fp – 5x5 Max Filter						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageMaxFilter5x5_Fp(mllib_image *dst, const mllib_image *src);</pre>						
DESCRIPTION	<p>The mllib_ImageMaxFilter5x5_Fp() function replaces the center pixel in a neighborhood with the floating-point maximum value in that neighborhood for each 5x5 neighborhood in the image.</p> <p>The source and destination images must be single-channel images.</p> <p>It uses the following equation:</p> $dst[x][y][0] = \text{MAX}\{ src[p][q][0], \quad x-2 \leq p \leq x+2; y-2 \leq q \leq y+2 \}$ <p>where $x = 2, \dots, w - 3$; $y = 2, \dots, h - 3$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<p>mllib_ImageMaxFilter3x3(3MLIB), mllib_ImageMaxFilter3x3_Fp(3MLIB), mllib_ImageMaxFilter5x5(3MLIB), mllib_ImageMaxFilter7x7(3MLIB), mllib_ImageMaxFilter7x7_Fp(3MLIB), mllib_ImageMedianFilter3x3(3MLIB), mllib_ImageMedianFilter3x3_Fp(3MLIB), mllib_ImageMedianFilter3x3_US(3MLIB), mllib_ImageMedianFilter5x5(3MLIB), mllib_ImageMedianFilter5x5_Fp(3MLIB), mllib_ImageMedianFilter5x5_US(3MLIB), mllib_ImageMedianFilter7x7(3MLIB), mllib_ImageMedianFilter7x7_Fp(3MLIB), mllib_ImageMedianFilter7x7_US(3MLIB), mllib_ImageMedianFilterMxN(3MLIB), mllib_ImageMedianFilterMxN_Fp(3MLIB), mllib_ImageMedianFilterMxN_US(3MLIB), mllib_ImageMinFilter3x3(3MLIB), mllib_ImageMinFilter3x3_Fp(3MLIB), mllib_ImageMinFilter5x5(3MLIB),</p>						

`mllib_ImageMaxFilter5x5_Fp(3MLIB)`

```
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7(3MLIB),  
mllib_ImageMinFilter7x7_Fp(3MLIB), mllib_ImageRankFilter3x3(3MLIB),  
mllib_ImageRankFilter3x3_Fp(3MLIB),  
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),  
mllib_ImageRankFilter5x5_Fp(3MLIB),  
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),  
mllib_ImageRankFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),  
mllib_ImageRankFilterMxN_Fp(3MLIB),  
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

mllib_ImageMaxFilter7x7(3MLIB)

NAME	mllib_ImageMaxFilter7x7 – 7x7 Max Filter						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageMaxFilter7x7(mllib_image *dst, const mllib_image *src);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageMaxFilter7x7()</code> function replaces the center pixel in a neighborhood with the maximum value in that neighborhood for each 7x7 neighborhood in the image.</p> <p>The source and destination images must be single-channel images.</p> <p>It uses the following equation:</p> $dst[x][y][0] = \text{MAX}\{ src[p][q][0], \quad x-3 \leq p \leq x+3; y-3 \leq q \leq y+3 \}$ <p>where $x = 3, \dots, w - 4$; $y = 3, \dots, h - 4$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<p><code>mllib_ImageMaxFilter3x3(3MLIB)</code>, <code>mllib_ImageMaxFilter3x3_Fp(3MLIB)</code>, <code>mllib_ImageMaxFilter5x5(3MLIB)</code>, <code>mllib_ImageMaxFilter5x5_Fp(3MLIB)</code>, <code>mllib_ImageMaxFilter7x7_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3_US(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5_US(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7_US(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN_US(3MLIB)</code>, <code>mllib_ImageMinFilter3x3(3MLIB)</code>, <code>mllib_ImageMinFilter3x3_Fp(3MLIB)</code>, <code>mllib_ImageMinFilter5x5(3MLIB)</code>,</p>						

`mllib_ImageMaxFilter7x7(3MLIB)`

```
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7(3MLIB),  
mllib_ImageMinFilter7x7_Fp(3MLIB), mllib_ImageRankFilter3x3(3MLIB),  
mllib_ImageRankFilter3x3_Fp(3MLIB),  
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),  
mllib_ImageRankFilter5x5_Fp(3MLIB),  
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),  
mllib_ImageRankFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),  
mllib_ImageRankFilterMxN_Fp(3MLIB),  
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

mllib_ImageMaxFilter7x7_Fp(3MLIB)

NAME	mllib_ImageMaxFilter7x7_Fp – 7x7 Max Filter						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageMaxFilter7x7_Fp(mllib_image *dst, const mllib_image *src);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageMaxFilter7x7_Fp()</code> function replaces the center pixel in a neighborhood with the floating-point maximum value in that neighborhood for each 7x7 neighborhood in the image.</p> <p>The source and destination images must be single-channel images.</p> <p>It uses the following equation:</p> $dst[x][y][0] = \text{MAX}\{ src[p][q][0], \quad x-3 \leq p \leq x+3; y-3 \leq q \leq y+3 \}$ <p>where $x = 3, \dots, w - 4$; $y = 3, \dots, h - 4$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageMaxFilter3x3(3MLIB)</code> , <code>mllib_ImageMaxFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMaxFilter5x5(3MLIB)</code> , <code>mllib_ImageMaxFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMaxFilter7x7(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3_US(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5_US(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7_US(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN_US(3MLIB)</code> , <code>mllib_ImageMinFilter3x3(3MLIB)</code> , <code>mllib_ImageMinFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMinFilter5x5(3MLIB)</code> ,						

`mllib_ImageMaxFilter7x7_Fp(3MLIB)`

```
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7(3MLIB),  
mllib_ImageMinFilter7x7_Fp(3MLIB), mllib_ImageRankFilter3x3(3MLIB),  
mllib_ImageRankFilter3x3_Fp(3MLIB),  
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),  
mllib_ImageRankFilter5x5_Fp(3MLIB),  
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),  
mllib_ImageRankFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),  
mllib_ImageRankFilterMxN_Fp(3MLIB),  
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

mllib_ImageMax_Fp(3MLIB)

NAME	mllib_ImageMax_Fp – two image maximum						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageMax_Fp(mllib_image *<i>dst</i>, const mllib_image *<i>src1</i>, const mllib_image *<i>src2</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageMax_Fp()</code> function accepts input from the two floating-point source images and writes the maximum to the destination image on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \text{MAX}\{ src1[x][y][i], src2[x][y][i] \}$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src1</i> Pointer to first source image.</p> <p><i>src2</i> Pointer to second source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageMax(3MLIB)</code> , <code>mllib_ImageMax_Fp_Inp(3MLIB)</code> , <code>mllib_ImageMax_Inp(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageMax_Fp_Inp – two image maximum

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageMax_Fp_Inp(mllib_image *src1dst, const
    mllib_image *src2);
```

DESCRIPTION The mllib_ImageMax_Fp_Inp() function accepts input from the two floating-point source images and writes the maximum, in place, on a pixel-by-pixel basis.

It uses the following equation:

$$src1dst[x][y][i] = \text{MAX}\{ src1dst[x][y][i], src2[x][y][i] \}$$

PARAMETERS The function takes the following arguments:

src1dst Pointer to source and destination image.

src2 Pointer to second source image.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_ImageMax\(3MLIB\)](#), [mllib_ImageMax_Fp\(3MLIB\)](#), [mllib_ImageMax_Inp\(3MLIB\)](#), [attributes\(5\)](#)

mllib_ImageMaximum(3MLIB)

NAME | mllib_ImageMaximum – image maximum

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageMaximum(mllib_s32 *max, const mllib_image
    *img) ;
```

DESCRIPTION | The `mllib_ImageMaximum()` function determines the maximum value for each channel in an image.

It uses the following equation:

$$\text{max}[i] = \text{MAX}\{ \text{img}[x][y][i]; \quad 0 \leq x < w, \quad 0 \leq y < h \}$$

PARAMETERS | The function takes the following arguments:

max | Pointer to maximum vector, where length is the number of channels in the image. `max[i]` contains the maximum of channel *i*.

img | Pointer to a source image.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageMaximum_Fp(3MLIB)`, `mllib_ImageMinimum(3MLIB)`, `mllib_ImageMinimum_Fp(3MLIB)`, `attributes(5)`

NAME mllib_ImageMaximum_Fp – image maximum

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageMaximum_Fp(mllib_d64 *max, const mllib_image
    *img);
```

DESCRIPTION The `mllib_ImageMaximum_Fp()` function determines the maximum value for each channel in a floating-point image.

It uses the following equation:

$$\max[i] = \text{MAX}\{ \text{img}[x][y][i]; 0 \leq x < w, 0 \leq y < h \}$$

PARAMETERS The function takes the following arguments:

max Pointer to maximum vector, where length is the number of channels in the image. `max[i]` contains the maximum of channel *i*.

img Pointer to a source image.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageMaximum(3MLIB)`, `mllib_ImageMinimum(3MLIB)`, `mllib_ImageMinimum_Fp(3MLIB)`, `attributes(5)`

mllib_ImageMax_Inp(3MLIB)

NAME	mllib_ImageMax_Inp – two image maximum						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageMax_Inp(mllib_image *<i>src1dst</i>, const mllib_image *<i>src2</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageMax_Inp()</code> function accepts input from the two source images and writes the maximum in place on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $\text{src1dst}[x][y][i] = \text{MAX}\{ \text{src1dst}[x][y][i], \text{src2}[x][y][i] \}$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>src1dst</i> Pointer to source and destination image.</p> <p><i>src2</i> Pointer to second source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageMax(3MLIB)</code> , <code>mllib_ImageMax_Fp(3MLIB)</code> , <code>mllib_ImageMax_Fp_Inp(3MLIB)</code> , <code>attributes(5)</code>						

NAME | mllib_ImageMean – image mean

SYNOPSIS | `cc [flag...] file... -lmllib [library...]`
`#include <mllib.h>`

`mllib_status mllib_ImageMean(mllib_d64 *mean, const mllib_image *img);`

DESCRIPTION | The mllib_ImageMean() function computes the mean value of all the pixels in the image.

It uses the following equation:

$$\text{mean}[i] = \frac{1}{w \cdot h} * \sum_{x=0}^{w-1} \sum_{y=0}^{h-1} \text{img}[x][y][i]$$

PARAMETERS | The function takes the following arguments:

mean | Pointer to mean array, where length is the number of channels in the image. mean[i] contains the mean of channel i.

img | Pointer to an image.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_ImageMean_Fp\(3MLIB\)](#), [mllib_ImageStdDev\(3MLIB\)](#), [mllib_ImageStdDev_Fp\(3MLIB\)](#), [attributes\(5\)](#)

mllib_ImageMean_Fp(3MLIB)

NAME | mllib_ImageMean_Fp – image mean

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageMean_Fp(mllib_d64 *mean, const mllib_image
    *img);
```

DESCRIPTION | The `mllib_ImageMean_Fp()` function computes the mean value of all the pixels in the image.

It uses the following equation:

$$\text{mean}[i] = \frac{1}{w \cdot h} * \sum_{x=0}^{w-1} \sum_{y=0}^{h-1} \text{img}[x][y][i]$$

PARAMETERS | The function takes the following arguments:

mean | Pointer to mean array, where length is the number of channels in the image. `mean[i]` contains the mean of channel `i`.

img | Pointer to an image.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageMean(3MLIB)`, `mllib_ImageStdDev(3MLIB)`, `mllib_ImageStdDev_Fp(3MLIB)`, `attributes(5)`

NAME mllib_ImageMedianFilter3x3 – 3x3 median filter

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageMedianFilter3x3(mllib_image *dst, const
    mllib_image *src, mllib_median_mask mmask, mllib_s32 cmask,
    mllib_edge edge);
```

DESCRIPTION The `mllib_ImageMedianFilter3x3()` function performs median filtering on an image. Each pixel of the destination image is the pixel with rank middle in the filter window.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

mmask Shape of the mask to be used for median filtering. It can be one of the following:

```
MLIB_MEDIAN_MASK_RECT
MLIB_MEDIAN_MASK_PLUS
MLIB_MEDIAN_MASK_X
MLIB_MEDIAN_MASK_RECT_SEPARABLE
```

cmask Channel mask to indicate the channels to be filtered. Each bit of which represents a channel in the image. The channels corresponded to 1 bits are those to be processed.

edge Type of edge condition. It can be one of the following:

```
MLIB_EDGE_DST_NO_WRITE
MLIB_EDGE_DST_FILL_ZERO
MLIB_EDGE_DST_COPY_SRC
MLIB_EDGE_SRC_EXTEND
```

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageMaxFilter3x3(3MLIB)`, `mllib_ImageMaxFilter3x3_Fp(3MLIB)`, `mllib_ImageMaxFilter5x5(3MLIB)`, `mllib_ImageMaxFilter5x5_Fp(3MLIB)`, `mllib_ImageMaxFilter7x7(3MLIB)`, `mllib_ImageMaxFilter7x7_Fp(3MLIB)`, `mllib_ImageMedianFilter3x3_Fp(3MLIB)`, `mllib_ImageMedianFilter3x3_US(3MLIB)`, `mllib_ImageMedianFilter5x5(3MLIB)`,

mllib_ImageMedianFilter3x3(3MLIB)

```
mllib_ImageMedianFilter5x5_Fp(3MLIB),
mllib_ImageMedianFilter5x5_US(3MLIB),
mllib_ImageMedianFilter7x7(3MLIB),
mllib_ImageMedianFilter7x7_Fp(3MLIB),
mllib_ImageMedianFilter7x7_US(3MLIB),
mllib_ImageMedianFilterMxN(3MLIB),
mllib_ImageMedianFilterMxN_Fp(3MLIB),
mllib_ImageMedianFilterMxN_US(3MLIB), mllib_ImageMinFilter3x3(3MLIB),
mllib_ImageMinFilter3x3_Fp(3MLIB), mllib_ImageMinFilter5x5(3MLIB),
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7(3MLIB),
mllib_ImageMinFilter7x7_Fp(3MLIB), mllib_ImageRankFilter3x3(3MLIB),
mllib_ImageRankFilter3x3_Fp(3MLIB),
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),
mllib_ImageRankFilter5x5_Fp(3MLIB),
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),
mllib_ImageRankFilter7x7_Fp(3MLIB),
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),
mllib_ImageRankFilterMxN_Fp(3MLIB),
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

NAME mllib_ImageMedianFilter3x3_Fp – 3x3 median filter

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageMedianFilter3x3_Fp(mllib_image *dst, const
    mllib_image *src, mllib_median_mask mmask, mllib_s32 cmask,
    mllib_edge edge);
```

DESCRIPTION The `mllib_ImageMedianFilter3x3_Fp()` function performs floating-point median filtering on an image. Each pixel of the destination image is the pixel with rank middle in the filter window.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

mmask Shape of the mask to be used for median filtering. It can be one of the following:

```
MLIB_MEDIAN_MASK_RECT
MLIB_MEDIAN_MASK_PLUS
MLIB_MEDIAN_MASK_X
MLIB_MEDIAN_MASK_RECT_SEPARABLE
```

cmask Channel mask to indicate the channels to be filtered. Each bit of which represents a channel in the image. The channels corresponded to 1 bits are those to be processed.

edge Type of edge condition. It can be one of the following:

```
MLIB_EDGE_DST_NO_WRITE
MLIB_EDGE_DST_FILL_ZERO
MLIB_EDGE_DST_COPY_SRC
MLIB_EDGE_SRC_EXTEND
```

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageMaxFilter3x3(3MLIB)`, `mllib_ImageMaxFilter3x3_Fp(3MLIB)`, `mllib_ImageMaxFilter5x5(3MLIB)`, `mllib_ImageMaxFilter5x5_Fp(3MLIB)`, `mllib_ImageMaxFilter7x7(3MLIB)`, `mllib_ImageMaxFilter7x7_Fp(3MLIB)`, `mllib_ImageMedianFilter3x3(3MLIB)`, `mllib_ImageMedianFilter3x3_US(3MLIB)`, `mllib_ImageMedianFilter5x5(3MLIB)`,

mllib_ImageMedianFilter3x3_Fp(3MLIB)

```
mllib_ImageMedianFilter5x5_Fp(3MLIB),  
mllib_ImageMedianFilter5x5_US(3MLIB),  
mllib_ImageMedianFilter7x7(3MLIB),  
mllib_ImageMedianFilter7x7_Fp(3MLIB),  
mllib_ImageMedianFilter7x7_US(3MLIB),  
mllib_ImageMedianFilterMxN(3MLIB),  
mllib_ImageMedianFilterMxN_Fp(3MLIB),  
mllib_ImageMedianFilterMxN_US(3MLIB), mllib_ImageMinFilter3x3(3MLIB),  
mllib_ImageMinFilter3x3_Fp(3MLIB), mllib_ImageMinFilter5x5(3MLIB),  
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7(3MLIB),  
mllib_ImageMinFilter7x7_Fp(3MLIB), mllib_ImageRankFilter3x3(3MLIB),  
mllib_ImageRankFilter3x3_Fp(3MLIB),  
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),  
mllib_ImageRankFilter5x5_Fp(3MLIB),  
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),  
mllib_ImageRankFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),  
mllib_ImageRankFilterMxN_Fp(3MLIB),  
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

NAME mllib_ImageMedianFilter3x3_US – 3x3 median filter, unsigned

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageMedianFilter3x3_US(mllib_image *dst, const
    mllib_image *src, mllib_median_mask mmask, mllib_s32 cmask,
    mllib_edge edge, mllib_s32 bits);
```

DESCRIPTION The `mllib_ImageMedianFilter3x3_US()` function performs median filtering on an `MLIB_SHORT` type of image that contains unsigned data. Each pixel of the destination image is the pixel with rank middle in the filter window.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

mmask Shape of the mask to be used for median filtering. It can be one of the following:

```
MLIB_MEDIAN_MASK_RECT
MLIB_MEDIAN_MASK_PLUS
MLIB_MEDIAN_MASK_X
MLIB_MEDIAN_MASK_RECT_SEPARABLE
```

cmask Channel mask to indicate the channels to be filtered. Each bit of which represents a channel in the image. The channels corresponded to 1 bits are those to be processed.

edge Type of edge condition. It can be one of the following:

```
MLIB_EDGE_DST_NO_WRITE
MLIB_EDGE_DST_FILL_ZERO
MLIB_EDGE_DST_COPY_SRC
MLIB_EDGE_SRC_EXTEND
```

bits The number of unsigned bits for pixel dynamic range. $9 \leq bits \leq 15$.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageMaxFilter3x3(3MLIB)`, `mllib_ImageMaxFilter3x3_Fp(3MLIB)`, `mllib_ImageMaxFilter5x5(3MLIB)`, `mllib_ImageMaxFilter5x5_Fp(3MLIB)`, `mllib_ImageMaxFilter7x7(3MLIB)`, `mllib_ImageMaxFilter7x7_Fp(3MLIB)`,

mllib_ImageMedianFilter3x3_US(3MLIB)

```
mllib_ImageMedianFilter3x3(3MLIB),
mllib_ImageMedianFilter3x3_Fp(3MLIB),
mllib_ImageMedianFilter5x5(3MLIB),
mllib_ImageMedianFilter5x5_Fp(3MLIB),
mllib_ImageMedianFilter5x5_US(3MLIB),
mllib_ImageMedianFilter7x7(3MLIB),
mllib_ImageMedianFilter7x7_Fp(3MLIB),
mllib_ImageMedianFilter7x7_US(3MLIB),
mllib_ImageMedianFilterMxN(3MLIB),
mllib_ImageMedianFilterMxN_Fp(3MLIB),
mllib_ImageMedianFilterMxN_US(3MLIB), mllib_ImageMinFilter3x3(3MLIB),
mllib_ImageMinFilter3x3_Fp(3MLIB), mllib_ImageMinFilter5x5(3MLIB),
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7(3MLIB),
mllib_ImageMinFilter7x7_Fp(3MLIB), mllib_ImageRankFilter3x3(3MLIB),
mllib_ImageRankFilter3x3_Fp(3MLIB),
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),
mllib_ImageRankFilter5x5_Fp(3MLIB),
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),
mllib_ImageRankFilter7x7_Fp(3MLIB),
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),
mllib_ImageRankFilterMxN_Fp(3MLIB),
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```


NAME mllib_ImageMedianFilter5x5 – 5x5 median filter

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageMedianFilter5x5(mllib_image *dst, const
    mllib_image *src, mllib_median_mask mmask, mllib_s32 cmask,
    mllib_edge edge);
```

DESCRIPTION The mllib_ImageMedianFilter5x5() function performs median filtering on an image. Each pixel of the destination image is the pixel with rank middle in the filter window.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

mmask Shape of the mask to be used for median filtering. It can be one of the following:

```
MLIB_MEDIAN_MASK_RECT
MLIB_MEDIAN_MASK_PLUS
MLIB_MEDIAN_MASK_X
MLIB_MEDIAN_MASK_RECT_SEPARABLE
```

cmask Channel mask to indicate the channels to be filtered. Each bit of which represents a channel in the image. The channels corresponded to 1 bits are those to be processed.

edge Type of edge condition. It can be one of the following:

```
MLIB_EDGE_DST_NO_WRITE
MLIB_EDGE_DST_FILL_ZERO
MLIB_EDGE_DST_COPY_SRC
MLIB_EDGE_SRC_EXTEND
```

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageMaxFilter3x3(3MLIB), mllib_ImageMaxFilter3x3_Fp(3MLIB), mllib_ImageMaxFilter5x5(3MLIB), mllib_ImageMaxFilter5x5_Fp(3MLIB), mllib_ImageMaxFilter7x7(3MLIB), mllib_ImageMaxFilter7x7_Fp(3MLIB), mllib_ImageMedianFilter3x3(3MLIB), mllib_ImageMedianFilter3x3_Fp(3MLIB), mllib_ImageMedianFilter3x3_US(3MLIB),

mllib_ImageMedianFilter5x5(3MLIB)

```
mllib_ImageMedianFilter5x5_Fp(3MLIB),
mllib_ImageMedianFilter5x5_US(3MLIB),
mllib_ImageMedianFilter7x7(3MLIB),
mllib_ImageMedianFilter7x7_Fp(3MLIB),
mllib_ImageMedianFilter7x7_US(3MLIB),
mllib_ImageMedianFilterMxN(3MLIB),
mllib_ImageMedianFilterMxN_Fp(3MLIB),
mllib_ImageMedianFilterMxN_US(3MLIB), mllib_ImageMinFilter3x3(3MLIB),
mllib_ImageMinFilter3x3_Fp(3MLIB), mllib_ImageMinFilter5x5(3MLIB),
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7(3MLIB),
mllib_ImageMinFilter7x7_Fp(3MLIB), mllib_ImageRankFilter3x3(3MLIB),
mllib_ImageRankFilter3x3_Fp(3MLIB),
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),
mllib_ImageRankFilter5x5_Fp(3MLIB),
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),
mllib_ImageRankFilter7x7_Fp(3MLIB),
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),
mllib_ImageRankFilterMxN_Fp(3MLIB),
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

NAME mllib_ImageMedianFilter5x5_Fp – 5x5 median filter

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageMedianFilter5x5_Fp(mllib_image *dst, const
mllib_image *src, mllib_median_mask mmask, mllib_s32 cmask,
mllib_edge edge);
```

DESCRIPTION The mllib_ImageMedianFilter5x5_Fp() function performs floating-point median filtering on an image. Each pixel of the destination image is the pixel with rank middle in the filter window.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

mmask Shape of the mask to be used for median filtering. It can be one of the following:

```
MLIB_MEDIAN_MASK_RECT
MLIB_MEDIAN_MASK_PLUS
MLIB_MEDIAN_MASK_X
MLIB_MEDIAN_MASK_RECT_SEPARABLE
```

cmask Channel mask to indicate the channels to be filtered. Each bit of which represents a channel in the image. The channels corresponded to 1 bits are those to be processed.

edge Type of edge condition. It can be one of the following:

```
MLIB_EDGE_DST_NO_WRITE
MLIB_EDGE_DST_FILL_ZERO
MLIB_EDGE_DST_COPY_SRC
MLIB_EDGE_SRC_EXTEND
```

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageMaxFilter3x3(3MLIB), mllib_ImageMaxFilter3x3_Fp(3MLIB), mllib_ImageMaxFilter5x5(3MLIB), mllib_ImageMaxFilter5x5_Fp(3MLIB), mllib_ImageMaxFilter7x7(3MLIB), mllib_ImageMaxFilter7x7_Fp(3MLIB), mllib_ImageMedianFilter3x3(3MLIB), mllib_ImageMedianFilter3x3_Fp(3MLIB), mllib_ImageMedianFilter3x3_US(3MLIB),

mllib_ImageMedianFilter5x5_Fp(3MLIB)

```
mllib_ImageMedianFilter5x5(3MLIB),
mllib_ImageMedianFilter5x5_US(3MLIB),
mllib_ImageMedianFilter7x7(3MLIB),
mllib_ImageMedianFilter7x7_Fp(3MLIB),
mllib_ImageMedianFilter7x7_US(3MLIB),
mllib_ImageMedianFilterMxN(3MLIB),
mllib_ImageMedianFilterMxN_Fp(3MLIB),
mllib_ImageMedianFilterMxN_US(3MLIB), mllib_ImageMinFilter3x3(3MLIB),
mllib_ImageMinFilter3x3_Fp(3MLIB), mllib_ImageMinFilter5x5(3MLIB),
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7(3MLIB),
mllib_ImageMinFilter7x7_Fp(3MLIB), mllib_ImageRankFilter3x3(3MLIB),
mllib_ImageRankFilter3x3_Fp(3MLIB),
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),
mllib_ImageRankFilter5x5_Fp(3MLIB),
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),
mllib_ImageRankFilter7x7_Fp(3MLIB),
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),
mllib_ImageRankFilterMxN_Fp(3MLIB),
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

NAME mllib_ImageMedianFilter5x5_US – 5x5 median filter, unsigned

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageMedianFilter5x5_US(mllib_image *dst, const
mllib_image *src, mllib_median_mask mmask, mllib_s32 cmask,
mllib_edge edge, mllib_s32 bits);
```

DESCRIPTION The mllib_ImageMedianFilter5x5_US() function performs median filtering on an MLIB_SHORT type of image that contains unsigned data. Each pixel of the destination image is the pixel with rank middle in the filter window.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

mmask Shape of the mask to be used for median filtering. It can be one of the following:

```
MLIB_MEDIAN_MASK_RECT
MLIB_MEDIAN_MASK_PLUS
MLIB_MEDIAN_MASK_X
MLIB_MEDIAN_MASK_RECT_SEPARABLE
```

cmask Channel mask to indicate the channels to be filtered. Each bit of which represents a channel in the image. The channels corresponded to 1 bits are those to be processed.

edge Type of edge condition. It can be one of the following:

```
MLIB_EDGE_DST_NO_WRITE
MLIB_EDGE_DST_FILL_ZERO
MLIB_EDGE_DST_COPY_SRC
MLIB_EDGE_SRC_EXTEND
```

bits The number of unsigned bits for pixel dynamic range. $9 \leq bits \leq 15$.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageMaxFilter3x3(3MLIB), mllib_ImageMaxFilter3x3_Fp(3MLIB), mllib_ImageMaxFilter5x5(3MLIB), mllib_ImageMaxFilter5x5_Fp(3MLIB), mllib_ImageMaxFilter7x7(3MLIB), mllib_ImageMaxFilter7x7_Fp(3MLIB),

mllib_ImageMedianFilter5x5_US(3MLIB)

```
mllib_ImageMedianFilter3x3(3MLIB),
mllib_ImageMedianFilter3x3_Fp(3MLIB),
mllib_ImageMedianFilter3x3_US(3MLIB),
mllib_ImageMedianFilter5x5(3MLIB),
mllib_ImageMedianFilter5x5_Fp(3MLIB),
mllib_ImageMedianFilter7x7(3MLIB),
mllib_ImageMedianFilter7x7_Fp(3MLIB),
mllib_ImageMedianFilter7x7_US(3MLIB),
mllib_ImageMedianFilterMxN(3MLIB),
mllib_ImageMedianFilterMxN_Fp(3MLIB),
mllib_ImageMedianFilterMxN_US(3MLIB), mllib_ImageMinFilter3x3(3MLIB),
mllib_ImageMinFilter3x3_Fp(3MLIB), mllib_ImageMinFilter5x5(3MLIB),
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7(3MLIB),
mllib_ImageMinFilter7x7_Fp(3MLIB), mllib_ImageRankFilter3x3(3MLIB),
mllib_ImageRankFilter3x3_Fp(3MLIB),
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),
mllib_ImageRankFilter5x5_Fp(3MLIB),
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),
mllib_ImageRankFilter7x7_Fp(3MLIB),
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),
mllib_ImageRankFilterMxN_Fp(3MLIB),
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

NAME mllib_ImageMedianFilter7x7 – 7x7 median filter

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageMedianFilter7x7(mllib_image *dst, const
    mllib_image *src, mllib_median_mask mmask, mllib_s32 cmask,
    mllib_edge edge);
```

DESCRIPTION The `mllib_ImageMedianFilter7x7()` function performs median filtering on an image. Each pixel of the destination image is the pixel with rank middle in the filter window.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

mmask Shape of the mask to be used for median filtering. It can be one of the following:

```
MLIB_MEDIAN_MASK_RECT
MLIB_MEDIAN_MASK_PLUS
MLIB_MEDIAN_MASK_X
MLIB_MEDIAN_MASK_RECT_SEPARABLE
```

cmask Channel mask to indicate the channels to be filtered. Each bit of which represents a channel in the image. The channels corresponded to 1 bits are those to be processed.

edge Type of edge condition. It can be one of the following:

```
MLIB_EDGE_DST_NO_WRITE
MLIB_EDGE_DST_FILL_ZERO
MLIB_EDGE_DST_COPY_SRC
MLIB_EDGE_SRC_EXTEND
```

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageMaxFilter3x3(3MLIB)`, `mllib_ImageMaxFilter3x3_Fp(3MLIB)`, `mllib_ImageMaxFilter5x5(3MLIB)`, `mllib_ImageMaxFilter5x5_Fp(3MLIB)`, `mllib_ImageMaxFilter7x7(3MLIB)`, `mllib_ImageMaxFilter7x7_Fp(3MLIB)`, `mllib_ImageMedianFilter3x3(3MLIB)`, `mllib_ImageMedianFilter3x3_Fp(3MLIB)`, `mllib_ImageMedianFilter3x3_US(3MLIB)`,

mllib_ImageMedianFilter7x7(3MLIB)

```
mllib_ImageMedianFilter5x5(3MLIB),
mllib_ImageMedianFilter5x5_Fp(3MLIB),
mllib_ImageMedianFilter5x5_US(3MLIB),
mllib_ImageMedianFilter7x7_Fp(3MLIB),
mllib_ImageMedianFilter7x7_US(3MLIB),
mllib_ImageMedianFilterMxN(3MLIB),
mllib_ImageMedianFilterMxN_Fp(3MLIB),
mllib_ImageMedianFilterMxN_US(3MLIB), mllib_ImageMinFilter3x3(3MLIB),
mllib_ImageMinFilter3x3_Fp(3MLIB), mllib_ImageMinFilter5x5(3MLIB),
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7(3MLIB),
mllib_ImageMinFilter7x7_Fp(3MLIB), mllib_ImageRankFilter3x3(3MLIB),
mllib_ImageRankFilter3x3_Fp(3MLIB),
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),
mllib_ImageRankFilter5x5_Fp(3MLIB),
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),
mllib_ImageRankFilter7x7_Fp(3MLIB),
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),
mllib_ImageRankFilterMxN_Fp(3MLIB),
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```


NAME mllib_ImageMedianFilter7x7_Fp – 7x7 median filter

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageMedianFilter7x7_Fp(mllib_image *dst, const
mllib_image *src, mllib_median_mask mmask, mllib_s32 cmask,
mllib_edge edge);
```

DESCRIPTION The `mllib_ImageMedianFilter7x7_Fp()` function performs floating-point median filtering on an image. Each pixel of the destination image is the pixel with rank middle in the filter window.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

mmask Shape of the mask to be used for median filtering. It can be one of the following:

```
MLIB_MEDIAN_MASK_RECT
MLIB_MEDIAN_MASK_PLUS
MLIB_MEDIAN_MASK_X
MLIB_MEDIAN_MASK_RECT_SEPARABLE
```

cmask Channel mask to indicate the channels to be filtered. Each bit of which represents a channel in the image. The channels corresponded to 1 bits are those to be processed.

edge Type of edge condition. It can be one of the following:

```
MLIB_EDGE_DST_NO_WRITE
MLIB_EDGE_DST_FILL_ZERO
MLIB_EDGE_DST_COPY_SRC
MLIB_EDGE_SRC_EXTEND
```

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageMaxFilter3x3(3MLIB)`, `mllib_ImageMaxFilter3x3_Fp(3MLIB)`, `mllib_ImageMaxFilter5x5(3MLIB)`, `mllib_ImageMaxFilter5x5_Fp(3MLIB)`, `mllib_ImageMaxFilter7x7(3MLIB)`, `mllib_ImageMaxFilter7x7_Fp(3MLIB)`, `mllib_ImageMedianFilter3x3(3MLIB)`, `mllib_ImageMedianFilter3x3_Fp(3MLIB)`, `mllib_ImageMedianFilter3x3_US(3MLIB)`,

mllib_ImageMedianFilter7x7_Fp(3MLIB)

```
mllib_ImageMedianFilter5x5(3MLIB),
mllib_ImageMedianFilter5x5_Fp(3MLIB),
mllib_ImageMedianFilter5x5_US(3MLIB),
mllib_ImageMedianFilter7x7(3MLIB),
mllib_ImageMedianFilter7x7_US(3MLIB),
mllib_ImageMedianFilterMxN(3MLIB),
mllib_ImageMedianFilterMxN_Fp(3MLIB),
mllib_ImageMedianFilterMxN_US(3MLIB), mllib_ImageMinFilter3x3(3MLIB),
mllib_ImageMinFilter3x3_Fp(3MLIB), mllib_ImageMinFilter5x5(3MLIB),
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7(3MLIB),
mllib_ImageMinFilter7x7_Fp(3MLIB), mllib_ImageRankFilter3x3(3MLIB),
mllib_ImageRankFilter3x3_Fp(3MLIB),
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),
mllib_ImageRankFilter5x5_Fp(3MLIB),
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),
mllib_ImageRankFilter7x7_Fp(3MLIB),
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),
mllib_ImageRankFilterMxN_Fp(3MLIB),
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

NAME mllib_ImageMedianFilter7x7_US – 7x7 median filter, unsigned

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageMedianFilter7x7_US(mllib_image *dst, const
    mllib_image *src, mllib_median_mask mmask, mllib_s32 cmask,
    mllib_edge edge, mllib_s32 bits);
```

DESCRIPTION The `mllib_ImageMedianFilter7x7_US()` function performs median filtering on an `MLIB_SHORT` type of image that contains unsigned data. Each pixel of the destination image is the pixel with rank middle in the filter window.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

mmask Shape of the mask to be used for median filtering. It can be one of the following:

```
MLIB_MEDIAN_MASK_RECT
MLIB_MEDIAN_MASK_PLUS
MLIB_MEDIAN_MASK_X
MLIB_MEDIAN_MASK_RECT_SEPARABLE
```

cmask Channel mask to indicate the channels to be filtered. Each bit of which represents a channel in the image. The channels corresponded to 1 bits are those to be processed.

edge Type of edge condition. It can be one of the following:

```
MLIB_EDGE_DST_NO_WRITE
MLIB_EDGE_DST_FILL_ZERO
MLIB_EDGE_DST_COPY_SRC
MLIB_EDGE_SRC_EXTEND
```

bits The number of unsigned bits for pixel dynamic range. $9 \leq bits \leq 15$.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageMaxFilter3x3(3MLIB)`, `mllib_ImageMaxFilter3x3_Fp(3MLIB)`, `mllib_ImageMaxFilter5x5(3MLIB)`, `mllib_ImageMaxFilter5x5_Fp(3MLIB)`, `mllib_ImageMaxFilter7x7(3MLIB)`, `mllib_ImageMaxFilter7x7_Fp(3MLIB)`,

mllib_ImageMedianFilter7x7_US(3MLIB)

```
mllib_ImageMedianFilter3x3(3MLIB),
mllib_ImageMedianFilter3x3_Fp(3MLIB),
mllib_ImageMedianFilter3x3_US(3MLIB),
mllib_ImageMedianFilter5x5(3MLIB),
mllib_ImageMedianFilter5x5_Fp(3MLIB),
mllib_ImageMedianFilter5x5_US(3MLIB),
mllib_ImageMedianFilter7x7(3MLIB),
mllib_ImageMedianFilter7x7_Fp(3MLIB),
mllib_ImageMedianFilterMxN(3MLIB),
mllib_ImageMedianFilterMxN_Fp(3MLIB),
mllib_ImageMedianFilterMxN_US(3MLIB), mllib_ImageMinFilter3x3(3MLIB),
mllib_ImageMinFilter3x3_Fp(3MLIB), mllib_ImageMinFilter5x5(3MLIB),
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7(3MLIB),
mllib_ImageMinFilter7x7_Fp(3MLIB), mllib_ImageRankFilter3x3(3MLIB),
mllib_ImageRankFilter3x3_Fp(3MLIB),
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),
mllib_ImageRankFilter5x5_Fp(3MLIB),
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),
mllib_ImageRankFilter7x7_Fp(3MLIB),
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),
mllib_ImageRankFilterMxN_Fp(3MLIB),
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

NAME mllib_ImageMedianFilterMxN – MxN median filter

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageMedianFilterMxN(mllib_image *dst, const
mllib_image *src, mllib_s32 m, mllib_s32 n, mllib_median_mask
mmask, mllib_s32 cmask, mllib_edge edge);
```

DESCRIPTION The mllib_ImageMedianFilterMxN() function performs MxN median filtering on an image. Each pixel of the destination image is the pixel with rank middle in the filter window.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

m Width of the filter window. *m* must be odd number greater than 1.

n Height of the filter window. *n* must be odd number greater than 1.

mmask Shape of the mask to be used for median filtering. It can be one of the following:

```
MLIB_MEDIAN_MASK_RECT
MLIB_MEDIAN_MASK_PLUS
MLIB_MEDIAN_MASK_X
MLIB_MEDIAN_MASK_RECT_SEPARABLE
```

cmask Channel mask to indicate the channels to be filtered. Each bit of which represents a channel in the image. The channels corresponded to 1 bits are those to be processed.

edge Type of edge condition. It can be one of the following:

```
MLIB_EDGE_DST_NO_WRITE
MLIB_EDGE_DST_FILL_ZERO
MLIB_EDGE_DST_COPY_SRC
MLIB_EDGE_SRC_EXTEND
```

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

mllib_ImageMedianFilterMxN(3MLIB)

SEE ALSO | mllib_ImageMaxFilter3x3(3MLIB), mllib_ImageMaxFilter3x3_Fp(3MLIB),
mllib_ImageMaxFilter5x5(3MLIB), mllib_ImageMaxFilter5x5_Fp(3MLIB),
mllib_ImageMaxFilter7x7(3MLIB), mllib_ImageMaxFilter7x7_Fp(3MLIB),
mllib_ImageMedianFilter3x3(3MLIB),
mllib_ImageMedianFilter3x3_Fp(3MLIB),
mllib_ImageMedianFilter3x3_US(3MLIB),
mllib_ImageMedianFilter5x5(3MLIB),
mllib_ImageMedianFilter5x5_Fp(3MLIB),
mllib_ImageMedianFilter5x5_US(3MLIB),
mllib_ImageMedianFilter7x7(3MLIB),
mllib_ImageMedianFilter7x7_Fp(3MLIB),
mllib_ImageMedianFilter7x7_US(3MLIB),
mllib_ImageMedianFilterMxN_Fp(3MLIB),
mllib_ImageMedianFilterMxN_US(3MLIB), mllib_ImageMinFilter3x3(3MLIB),
mllib_ImageMinFilter3x3_Fp(3MLIB), mllib_ImageMinFilter5x5(3MLIB),
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7(3MLIB),
mllib_ImageMinFilter7x7_Fp(3MLIB), mllib_ImageRankFilter3x3(3MLIB),
mllib_ImageRankFilter3x3_Fp(3MLIB),
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),
mllib_ImageRankFilter5x5_Fp(3MLIB),
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),
mllib_ImageRankFilter7x7_Fp(3MLIB),
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),
mllib_ImageRankFilterMxN_Fp(3MLIB),
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)

mllib_ImageMedianFilterMxN_Fp(3MLIB)

NAME mllib_ImageMedianFilterMxN_Fp – MxN median filter

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageMedianFilterMxN_Fp(mllib_image *dst, const
mllib_image *src, mllib_s32 m, mllib_s32 n, mllib_median_mask
mmask, mllib_s32 cmask, mllib_edge edge);
```

DESCRIPTION The `mllib_ImageMedianFilterMxN_Fp()` function performs MxN median filtering on a floating-point image. Each pixel of the destination image is the pixel with rank middle in the filter window.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

m Width of the filter window. *m* must be odd number greater than 1.

n Height of the filter window. *n* must be odd number greater than 1.

mmask Shape of the mask to be used for median filtering. It can be one of the following:

```
MLIB_MEDIAN_MASK_RECT
MLIB_MEDIAN_MASK_PLUS
MLIB_MEDIAN_MASK_X
MLIB_MEDIAN_MASK_RECT_SEPARABLE
```

cmask Channel mask to indicate the channels to be filtered. Each bit of which represents a channel in the image. The channels corresponded to 1 bits are those to be processed.

edge Type of edge condition. It can be one of the following:

```
MLIB_EDGE_DST_NO_WRITE
MLIB_EDGE_DST_FILL_ZERO
MLIB_EDGE_DST_COPY_SRC
MLIB_EDGE_SRC_EXTEND
```

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

mllib_ImageMedianFilterMxN_Fp(3MLIB)

SEE ALSO | mllib_ImageMaxFilter3x3(3MLIB), mllib_ImageMaxFilter3x3_Fp(3MLIB),
mllib_ImageMaxFilter5x5(3MLIB), mllib_ImageMaxFilter5x5_Fp(3MLIB),
mllib_ImageMaxFilter7x7(3MLIB), mllib_ImageMaxFilter7x7_Fp(3MLIB),
mllib_ImageMedianFilter3x3(3MLIB),
mllib_ImageMedianFilter3x3_Fp(3MLIB),
mllib_ImageMedianFilter3x3_US(3MLIB),
mllib_ImageMedianFilter5x5(3MLIB),
mllib_ImageMedianFilter5x5_Fp(3MLIB),
mllib_ImageMedianFilter5x5_US(3MLIB),
mllib_ImageMedianFilter7x7(3MLIB),
mllib_ImageMedianFilter7x7_Fp(3MLIB),
mllib_ImageMedianFilter7x7_US(3MLIB),
mllib_ImageMedianFilterMxN(3MLIB),
mllib_ImageMedianFilterMxN_US(3MLIB), mllib_ImageMinFilter3x3(3MLIB),
mllib_ImageMinFilter3x3_Fp(3MLIB), mllib_ImageMinFilter5x5(3MLIB),
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7(3MLIB),
mllib_ImageMinFilter7x7_Fp(3MLIB), mllib_ImageRankFilter3x3(3MLIB),
mllib_ImageRankFilter3x3_Fp(3MLIB),
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),
mllib_ImageRankFilter5x5_Fp(3MLIB),
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),
mllib_ImageRankFilter7x7_Fp(3MLIB),
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),
mllib_ImageRankFilterMxN_Fp(3MLIB),
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)

NAME mllib_ImageMedianFilterMxN_US – MxN median filter, unsigned

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageMedianFilterMxN_US(mllib_image *dst, const
mllib_image *src, mllib_s32 m, mllib_s32 n, mllib_median_mask
mmask, mllib_s32 cmask, mllib_edge edge, mllib_s32 bits);
```

DESCRIPTION The `mllib_ImageMedianFilterMxN_US()` function performs MxN median filtering on an `MLIB_SHORT` type of image that contains unsigned data. Each pixel of the destination image is the pixel with rank middle in the filter window.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

m Width of the filter window. *m* must be odd number greater than 1.

n Height of the filter window. *n* must be odd number greater than 1.

mmask Shape of the mask to be used for median filtering. It can be one of the following:

```
MLIB_MEDIAN_MASK_RECT
MLIB_MEDIAN_MASK_PLUS
MLIB_MEDIAN_MASK_X
MLIB_MEDIAN_MASK_RECT_SEPARABLE
```

cmask Channel mask to indicate the channels to be filtered. Each bit of which represents a channel in the image. The channels corresponded to 1 bits are those to be processed.

edge Type of edge condition. It can be one of the following:

```
MLIB_EDGE_DST_NO_WRITE
MLIB_EDGE_DST_FILL_ZERO
MLIB_EDGE_DST_COPY_SRC
MLIB_EDGE_SRC_EXTEND
```

bits The number of unsigned bits for pixel dynamic range. $9 \leq bits \leq 15$.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

mllib_ImageMedianFilterMxN_US(3MLIB)

SEE ALSO | mllib_ImageMaxFilter3x3(3MLIB), mllib_ImageMaxFilter3x3_Fp(3MLIB),
mllib_ImageMaxFilter5x5(3MLIB), mllib_ImageMaxFilter5x5_Fp(3MLIB),
mllib_ImageMaxFilter7x7(3MLIB), mllib_ImageMaxFilter7x7_Fp(3MLIB),
mllib_ImageMedianFilter3x3(3MLIB),
mllib_ImageMedianFilter3x3_Fp(3MLIB),
mllib_ImageMedianFilter3x3_US(3MLIB),
mllib_ImageMedianFilter5x5(3MLIB),
mllib_ImageMedianFilter5x5_Fp(3MLIB),
mllib_ImageMedianFilter5x5_US(3MLIB),
mllib_ImageMedianFilter7x7(3MLIB),
mllib_ImageMedianFilter7x7_Fp(3MLIB),
mllib_ImageMedianFilter7x7_US(3MLIB),
mllib_ImageMedianFilterMxN(3MLIB),
mllib_ImageMedianFilterMxN_Fp(3MLIB), mllib_ImageMinFilter3x3(3MLIB),
mllib_ImageMinFilter3x3_Fp(3MLIB), mllib_ImageMinFilter5x5(3MLIB),
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7(3MLIB),
mllib_ImageMinFilter7x7_Fp(3MLIB), mllib_ImageRankFilter3x3(3MLIB),
mllib_ImageRankFilter3x3_Fp(3MLIB),
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),
mllib_ImageRankFilter5x5_Fp(3MLIB),
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),
mllib_ImageRankFilter7x7_Fp(3MLIB),
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),
mllib_ImageRankFilterMxN_Fp(3MLIB),
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)

NAME mlib_ImageMin – two-image minimum

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_ImageMin(mlib_image *dst, const mlib_image *src1,
    const mlib_image *src2);
```

DESCRIPTION The `mlib_ImageMin()` function accepts input from the two source images and writes the minimum to the destination image on a pixel-by-pixel basis.

It uses the following equation:

$$dst[x][y][i] = \text{MIN}\{ src1[x][y][i], src2[x][y][i] \}$$

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src1 Pointer to first source image.

src2 Pointer to second source image.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mlib_ImageMin_Fp(3MLIB)`, `mlib_ImageMin_Fp_Inp(3MLIB)`, `mlib_ImageMin_Inp(3MLIB)`, `attributes(5)`

mllib_ImageMinFilter3x3(3MLIB)

NAME	mllib_ImageMinFilter3x3 – 3x3 Min Filter						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageMinFilter3x3(mllib_image *dst, const mllib_image *src);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageMinFilter3x3()</code> function replaces the center pixel in a neighborhood with the minimum value in that neighborhood for each 3x3 neighborhood in the image.</p> <p>The source and destination images must be single-channel images.</p> <p>It uses the following equation:</p> $dst[x][y][0] = \text{MIN}\{ src[p][q][0], \quad x-1 \leq p \leq x+1; y-1 \leq q \leq y+1 \}$ <p>where $x = 1, \dots, w - 2$; $y = 1, \dots, h - 2$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<p><code>mllib_ImageMaxFilter3x3(3MLIB)</code>, <code>mllib_ImageMaxFilter3x3_Fp(3MLIB)</code>, <code>mllib_ImageMaxFilter5x5(3MLIB)</code>, <code>mllib_ImageMaxFilter5x5_Fp(3MLIB)</code>, <code>mllib_ImageMaxFilter7x7(3MLIB)</code>, <code>mllib_ImageMaxFilter7x7_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3_US(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5_US(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7_US(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN_US(3MLIB)</code>,</p>						

mllib_ImageMinFilter3x3(3MLIB)

```
mllib_ImageMinFilter3x3_Fp(3MLIB), mllib_ImageMinFilter5x5(3MLIB),  
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7(3MLIB),  
mllib_ImageMinFilter7x7_Fp(3MLIB), mllib_ImageRankFilter3x3(3MLIB),  
mllib_ImageRankFilter3x3_Fp(3MLIB),  
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),  
mllib_ImageRankFilter5x5_Fp(3MLIB),  
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),  
mllib_ImageRankFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),  
mllib_ImageRankFilterMxN_Fp(3MLIB),  
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

mllib_ImageMinFilter3x3_Fp(3MLIB)

NAME	mllib_ImageMinFilter3x3_Fp – 3x3 Min Filter, floating point						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageMinFilter3x3_Fp(mllib_image *dst, const mllib_image *src);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageMinFilter3x3_Fp()</code> function replaces the center pixel in a neighborhood with the floating-point minimum value in that neighborhood for each 3x3 neighborhood in the image.</p> <p>The source and destination images must be single-channel images.</p> <p>It uses the following equation:</p> $dst[x][y][0] = \text{MIN}\{ src[p][q][0], \quad x-1 \leq p \leq x+1; y-1 \leq q \leq y+1 \}$ <p>where $x = 1, \dots, w - 2$; $y = 1, \dots, h - 2$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<p><code>mllib_ImageMaxFilter3x3(3MLIB)</code>, <code>mllib_ImageMaxFilter3x3_Fp(3MLIB)</code>, <code>mllib_ImageMaxFilter5x5(3MLIB)</code>, <code>mllib_ImageMaxFilter5x5_Fp(3MLIB)</code>, <code>mllib_ImageMaxFilter7x7(3MLIB)</code>, <code>mllib_ImageMaxFilter7x7_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3_US(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5_US(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7_US(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN_US(3MLIB)</code>, <code>mllib_ImageMinFilter3x3(3MLIB)</code>,</p>						

mllib_ImageMinFilter3x3_Fp(3MLIB)

```
mllib_ImageMinFilter5x5(3MLIB), mllib_ImageMinFilter5x5_Fp(3MLIB),  
mllib_ImageMinFilter7x7(3MLIB), mllib_ImageMinFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter3x3(3MLIB), mllib_ImageRankFilter3x3_Fp(3MLIB),  
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),  
mllib_ImageRankFilter5x5_Fp(3MLIB),  
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),  
mllib_ImageRankFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),  
mllib_ImageRankFilterMxN_Fp(3MLIB),  
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

mllib_ImageMinFilter5x5(3MLIB)

NAME	mllib_ImageMinFilter5x5 – 5x5 Min Filter						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageMinFilter5x5(mllib_image *dst, const mllib_image *src);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageMinFilter5x5()</code> function replaces the center pixel in a neighborhood with the minimum value in that neighborhood for each 5x5 neighborhood in the image.</p> <p>The source and destination images must be single-channel images.</p> <p>It uses the following equation:</p> $dst[x][y][0] = \text{MIN}\{ src[p][q][0], \quad x-2 \leq p \leq x+2; y-2 \leq q \leq y+2 \}$ <p>where $x = 2, \dots, w - 3$; $y = 2, \dots, h - 3$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<p><code>mllib_ImageMaxFilter3x3(3MLIB)</code>, <code>mllib_ImageMaxFilter3x3_Fp(3MLIB)</code>, <code>mllib_ImageMaxFilter5x5(3MLIB)</code>, <code>mllib_ImageMaxFilter5x5_Fp(3MLIB)</code>, <code>mllib_ImageMaxFilter7x7(3MLIB)</code>, <code>mllib_ImageMaxFilter7x7_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3_US(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5_US(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7_US(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN_US(3MLIB)</code>, <code>mllib_ImageMinFilter3x3(3MLIB)</code>,</p>						

mllib_ImageMinFilter5x5(3MLIB)

```
mllib_ImageMinFilter3x3_Fp(3MLIB), mllib_ImageMinFilter5x5_Fp(3MLIB),  
mllib_ImageMinFilter7x7(3MLIB), mllib_ImageMinFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter3x3(3MLIB), mllib_ImageRankFilter3x3_Fp(3MLIB),  
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),  
mllib_ImageRankFilter5x5_Fp(3MLIB),  
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),  
mllib_ImageRankFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),  
mllib_ImageRankFilterMxN_Fp(3MLIB),  
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

mllib_ImageMinFilter5x5_Fp(3MLIB)

NAME	mllib_ImageMinFilter5x5_Fp – 5x5 Min Filter, floating point						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageMinFilter5x5_Fp(mllib_image *dst, const mllib_image *src);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageMinFilter5x5_Fp()</code> function replaces the center pixel in a neighborhood with the floating-point minimum value in that neighborhood for each 5x5 neighborhood in the image.</p> <p>The source and destination images must be single-channel images.</p> <p>It uses the following equation:</p> $dst[x][y][0] = \text{MIN}\{ src[p][q][0], \quad x-2 \leq p \leq x+2; y-2 \leq q \leq y+2 \}$ <p>where $x = 2, \dots, w - 3$; $y = 2, \dots, h - 3$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageMaxFilter3x3(3MLIB)</code> , <code>mllib_ImageMaxFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMaxFilter5x5(3MLIB)</code> , <code>mllib_ImageMaxFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMaxFilter7x7(3MLIB)</code> , <code>mllib_ImageMaxFilter7x7_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3_US(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5_US(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7_US(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN_US(3MLIB)</code> , <code>mllib_ImageMinFilter3x3(3MLIB)</code> ,						

mllib_ImageMinFilter5x5_Fp(3MLIB)

```
mllib_ImageMinFilter3x3_Fp(3MLIB), mllib_ImageMinFilter5x5(3MLIB),  
mllib_ImageMinFilter7x7(3MLIB), mllib_ImageMinFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter3x3(3MLIB), mllib_ImageRankFilter3x3_Fp(3MLIB),  
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),  
mllib_ImageRankFilter5x5_Fp(3MLIB),  
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),  
mllib_ImageRankFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),  
mllib_ImageRankFilterMxN_Fp(3MLIB),  
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

mllib_ImageMinFilter7x7(3MLIB)

NAME	mllib_ImageMinFilter7x7 – 7x7 Min Filter						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageMinFilter7x7(mllib_image *dst, const mllib_image *src);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageMinFilter7x7()</code> function replaces the center pixel in a neighborhood with the minimum value in that neighborhood for each 7x7 neighborhood in the image.</p> <p>The source and destination images must be single-channel images.</p> <p>It uses the following equation:</p> $dst[x][y][0] = \text{MIN}\left\{ \begin{array}{l} src[p][q][0], \\ x-3 \leq p \leq x+3; y-3 \leq q \leq y+3 \end{array} \right\}$ <p>where $x = 3, \dots, w - 4$; $y = 3, \dots, h - 4$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageMaxFilter3x3(3MLIB)</code> , <code>mllib_ImageMaxFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMaxFilter5x5(3MLIB)</code> , <code>mllib_ImageMaxFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMaxFilter7x7(3MLIB)</code> , <code>mllib_ImageMaxFilter7x7_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3_US(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5_US(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7_US(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN_US(3MLIB)</code> , <code>mllib_ImageMinFilter3x3(3MLIB)</code> ,						

mllib_ImageMinFilter7x7(3MLIB)

```
mllib_ImageMinFilter3x3_Fp(3MLIB), mllib_ImageMinFilter5x5(3MLIB),  
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter3x3(3MLIB), mllib_ImageRankFilter3x3_Fp(3MLIB),  
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),  
mllib_ImageRankFilter5x5_Fp(3MLIB),  
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),  
mllib_ImageRankFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),  
mllib_ImageRankFilterMxN_Fp(3MLIB),  
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

mllib_ImageMinFilter7x7_Fp(3MLIB)

NAME	mllib_ImageMinFilter7x7_Fp – 7x7 Min Filter, floating point						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageMinFilter7x7_Fp(mllib_image *dst, const mllib_image *src);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageMinFilter7x7_Fp()</code> function replaces the center pixel in a neighborhood with the floating-point minimum value in that neighborhood for each 7x7 neighborhood in the image.</p> <p>The source and destination images must be single-channel images.</p> <p>It uses the following equation:</p> $dst[x][y][0] = \text{MIN}\{ src[p][q][0], x-3 \leq p \leq x+3; y-3 \leq q \leq y+3 \}$ <p>where $x = 3, \dots, w - 4$; $y = 3, \dots, h - 4$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageMaxFilter3x3(3MLIB)</code> , <code>mllib_ImageMaxFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMaxFilter5x5(3MLIB)</code> , <code>mllib_ImageMaxFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMaxFilter7x7(3MLIB)</code> , <code>mllib_ImageMaxFilter7x7_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3_US(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5_US(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7_US(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN_US(3MLIB)</code> , <code>mllib_ImageMinFilter3x3(3MLIB)</code> ,						

mllib_ImageMinFilter7x7_Fp(3MLIB)

```
mllib_ImageMinFilter3x3_Fp(3MLIB), mllib_ImageMinFilter5x5(3MLIB),  
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7(3MLIB),  
mllib_ImageRankFilter3x3(3MLIB), mllib_ImageRankFilter3x3_Fp(3MLIB),  
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),  
mllib_ImageRankFilter5x5_Fp(3MLIB),  
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),  
mllib_ImageRankFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),  
mllib_ImageRankFilterMxN_Fp(3MLIB),  
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

mllib_ImageMin_Fp(3MLIB)

NAME | mllib_ImageMin_Fp – two-image minimum

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageMin_Fp(mllib_image *dst, const mllib_image
    *src1, const mllib_image *src2);
```

DESCRIPTION | The `mllib_ImageMin_Fp()` function accepts input from the two floating-point source images and writes the minimum to the destination image on a pixel-by-pixel basis.

It uses the following equation:

$$dst[x][y][i] = \text{MIN}\{ src1[x][y][i], src2[x][y][i] \}$$

PARAMETERS | The function takes the following arguments:

dst | Pointer to destination image.

src1 | Pointer to first source image.

src2 | Pointer to second source image.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageMin(3MLIB)`, `mllib_ImageMin_Fp_Inp(3MLIB)`,
`mllib_ImageMin_Inp(3MLIB)`, `attributes(5)`

NAME mllib_ImageMin_Fp_Inp – two-image minimum

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageMin_Fp_Inp(mllib_image *src1dst, const
    mllib_image *src2);
```

DESCRIPTION The mllib_ImageMin_Fp_Inp() function accepts input from the two source images and writes the minimum in place on a pixel-by-pixel basis.

It uses the following equation:

$$src1dst[x][y][i] = \text{MIN}\{ src1dst[x][y][i], src2[x][y][i] \}$$

PARAMETERS The function takes the following arguments:

src1dst Pointer to source and destination image.

src2 Pointer to second source image.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_ImageMin\(3MLIB\)](#), [mllib_ImageMin_Fp\(3MLIB\)](#), [mllib_ImageMin_Inp\(3MLIB\)](#), [attributes\(5\)](#)

mllib_ImageMinimum(3MLIB)

NAME | mllib_ImageMinimum – image minimum

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageMinimum(mllib_s32 *min, const mllib_image
    *img);
```

DESCRIPTION | The mllib_ImageMinimum() function determines the minimum value for each channel in an image.

It uses the following equation:

$$\text{min}[i] = \text{MIN}\{ \text{img}[x][y][i]; \quad 0 \leq x < w, \quad 0 \leq y < h \}$$

PARAMETERS | The function takes the following arguments:

min | Pointer to minimum vector, where length is the number of channels in the image. min[i] contains the minimum of channel i.

img | Pointer to a source image.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_ImageMaximum(3MLIB), mllib_ImageMaximum_Fp(3MLIB), mllib_ImageMinimum_Fp(3MLIB), attributes(5)

NAME	mllib_ImageMinimum_Fp – image minimum						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageMinimum_Fp(mllib_d64 *min, const mllib_image *img);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageMinimum_Fp()</code> function determines the minimum value for each channel in a floating-point image.</p> <p>It uses the following equation:</p> $\text{min}[i] = \text{MIN}\{ \text{img}[x][y][i]; 0 \leq x < w, 0 \leq y < h \}$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>min</i> Pointer to minimum vector, where length is the number of channels in the image. <code>min[i]</code> contains the minimum of channel <i>i</i>.</p> <p><i>img</i> Pointer to a source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageMaximum(3MLIB)</code> , <code>mllib_ImageMaximum_Fp(3MLIB)</code> , <code>mllib_ImageMinimum(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageMin_Inp(3MLIB)

NAME | mllib_ImageMin_Inp – two-image minimum

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageMin_Inp(mllib_image *src1dst, const mllib_image
*src2) ;
```

DESCRIPTION | The `mllib_ImageMin_Inp()` function accepts input from the two source images and writes the minimum in place on a pixel-by-pixel basis.

It uses the following equation:

$$\text{src1dst}[x][y][i] = \text{MIN}\{ \text{src1dst}[x][y][i], \text{src2}[x][y][i] \}$$

PARAMETERS | The function takes the following arguments:

src1dst Pointer to source and destination image.

src2 Pointer to second source image.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageMin(3MLIB)`, `mllib_ImageMin_Fp(3MLIB)`, `mllib_ImageMin_Fp_Inp(3MLIB)`, `attributes(5)`

NAME	mllib_ImageMoment2 – second moment						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageMoment2(mllib_d64 *moment, const mllib_image *img);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageMoment2()</code> function computes the second moment of each channel in an image.</p> <p>It uses the following equation:</p> $\text{moment}[i] = \frac{1}{w \cdot h} * \sum_{x=0}^{w-1} \sum_{y=0}^{h-1} \text{img}[x][y][i]**2$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>moment</i> Pointer to moment array, where length is the number of channels in the image.</p> <p><i>img</i> Pointer to an image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageMoment2_Fp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageMoment2_Fp(3MLIB)

NAME | mllib_ImageMoment2_Fp – second moment

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageMoment2_Fp(mllib_d64 *moment, const
    mllib_image *img);
```

DESCRIPTION | The `mllib_ImageMoment2_Fp()` function computes the second moment of each channel in a floating-point image.

It uses the following equation:

$$\text{moment}[i] = \frac{1}{w \cdot h} * \sum_{x=0}^{w-1} \sum_{y=0}^{h-1} \text{img}[x][y][i]**2$$

PARAMETERS | The function takes the following arguments:

moment | Pointer to moment array, where length is the number of channels in the image.

img | Pointer to an image.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_ImageMoment2\(3MLIB\)](#), [attributes\(5\)](#)

NAME	mllib_ImageMulAlpha – alpha channel multiplication						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageMulAlpha(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, mllib_s32 <i>cmask</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageMulAlpha()</code> function multiplies color channels by the alpha channel on a pixel by pixel basis.</p> <p>For the <code>MLIB_BYTE</code> image, it uses the following equation:</p> $dst[x][y][c] = src[x][y][c] * src[x][y][a] * 2^{*(-8)}$ <p>For the <code>MLIB_SHORT</code> image, it uses the following equation:</p> $dst[x][y][c] = src[x][y][c] * src[x][y][a] * 2^{*(-15)}$ <p>For the <code>MLIB_USHORT</code> image, it uses the following equation:</p> $dst[x][y][c] = src[x][y][c] * src[x][y][a] * 2^{*(-16)}$ <p>For the <code>MLIB_INT</code> image, it uses the following equation:</p> $dst[x][y][c] = src[x][y][c] * src[x][y][a] * 2^{*(-31)}$ <p>where <code>c</code> and <code>a</code> are the indices for the color channels and the alpha channel, respectively, so <code>c != a</code>.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>cmask</i> Channel mask to indicate the alpha channel. Each bit of the mask represents a channel in the image. The channel corresponding to the 1 bit of <code>cmask</code> is the alpha channel.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageMulAlpha_Inp(3MLIB)</code> , <code>mllib_ImageMulAlpha_Fp(3MLIB)</code> , <code>mllib_ImageMulAlpha_Fp_Inp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageMulAlpha_Fp(3MLIB)

NAME	mllib_ImageMulAlpha_Fp – alpha channel multiplication						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageMulAlpha_Fp(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, mllib_s32 <i>cmask</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageMulAlpha_Fp()</code> function multiplies floating-point color channels by the alpha channel on a pixel by pixel basis.</p> <p>It uses the following equation:</p> $dst[x][y][c] = src[x][y][c] * src[x][y][a]$ <p>where <i>c</i> and <i>a</i> are the indices for the color channels and the alpha channel, respectively, so $c \neq a$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>cmask</i> Channel mask to indicate the alpha channel. Each bit of the mask represents a channel in the image. The channel corresponding to the 1 bit of <i>cmask</i> is the alpha channel.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageMulAlpha(3MLIB)</code> , <code>mllib_ImageMulAlpha_Inp(3MLIB)</code> , <code>mllib_ImageMulAlpha_Fp_Inp(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageMulAlpha_Fp_Inp – alpha channel multiplication

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>
```

```
mllib_status mllib_ImageMulAlpha_Fp_Inp(mllib_image *srcdst, mllib_s32
    cmask);
```

DESCRIPTION The mllib_ImageMulAlpha_Fp_Inp() function multiplies floating-point color channels by the alpha channel on a pixel by pixel basis, in place.

It uses the following equation:

$$srcdst[x][y][c] = srcdst[x][y][c] * srcdst[x][y][a]$$

where c and a are the indices for the color channels and the alpha channel, respectively, so c != a.

PARAMETERS The function takes the following arguments:

- srcdst* Pointer to source and destination image.
- cmask* Channel mask to indicate the alpha channel. Each bit of the mask represents a channel in the image. The channel corresponding to the 1 bit of *cmask* is the alpha channel.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_ImageMulAlpha\(3MLIB\)](#), [mllib_ImageMulAlpha_Inp\(3MLIB\)](#), [mllib_ImageMulAlpha_Fp\(3MLIB\)](#), [attributes\(5\)](#)

mllib_ImageMulAlpha_Inp(3MLIB)

NAME	mllib_ImageMulAlpha_Inp – alpha channel multiplication, in place						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageMulAlpha_Inp(mllib_image *srcdst, mllib_s32 cmask);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageMulAlpha_Inp()</code> function multiplies color channels by the alpha channel on a pixel by pixel basis, in place.</p> <p>For the <code>MLIB_BYTE</code> image, it uses the following equation:</p> $\text{srcdst}[x][y][c] = \text{srcdst}[x][y][c] * \text{srcdst}[x][y][a] * 2^{**(-8)}$ <p>For the <code>MLIB_SHORT</code> image, it uses the following equation:</p> $\text{srcdst}[x][y][c] = \text{srcdst}[x][y][c] * \text{srcdst}[x][y][a] * 2^{**(-15)}$ <p>For the <code>MLIB_USHORT</code> image, it uses the following equation:</p> $\text{srcdst}[x][y][c] = \text{srcdst}[x][y][c] * \text{srcdst}[x][y][a] * 2^{**(-16)}$ <p>For the <code>MLIB_INT</code> image, it uses the following equation:</p> $\text{srcdst}[x][y][c] = \text{srcdst}[x][y][c] * \text{srcdst}[x][y][a] * 2^{**(-31)}$ <p>where <code>c</code> and <code>a</code> are the indices for the color channels and the alpha channel, respectively, so <code>c != a</code>.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>srcdst</i></td><td>Pointer to source and destination image.</td></tr><tr><td><i>cmask</i></td><td>Channel mask to indicate the alpha channel. Each bit of the mask represents a channel in the image. The channel corresponding to the 1 bit of <i>cmask</i> is the alpha channel.</td></tr></table>	<i>srcdst</i>	Pointer to source and destination image.	<i>cmask</i>	Channel mask to indicate the alpha channel. Each bit of the mask represents a channel in the image. The channel corresponding to the 1 bit of <i>cmask</i> is the alpha channel.		
<i>srcdst</i>	Pointer to source and destination image.						
<i>cmask</i>	Channel mask to indicate the alpha channel. Each bit of the mask represents a channel in the image. The channel corresponding to the 1 bit of <i>cmask</i> is the alpha channel.						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageMulAlpha(3MLIB)</code> , <code>mllib_ImageMulAlpha_Fp(3MLIB)</code> , <code>mllib_ImageMulAlpha_Fp_Inp(3MLIB)</code> , <code>attributes(5)</code>						

NAME	mllib_ImageMul_Fp – computes the multiplication of two images on a pixel-by-pixel basis						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmlib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageMul_Fp(mllib_image <i>dst</i>, const mllib_image *<i>src1</i>, const mllib_image *<i>src2</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageMul_Fp()</code> function computes the multiplication of two floating-point images on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $dst[x][y][i] = src1[x][y][i] * src2[x][y][i]$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src1</i> Pointer to first source image.</p> <p><i>src2</i> Pointer to second source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageMul_Fp_Inp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageMul_Fp_Inp(3MLIB)

NAME | mllib_ImageMul_Fp_Inp – computes the multiplication of two images on a pixel-by-pixel basis

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageMul_Fp_Inp(mllib_image *src1dst, const
mllib_image *src2);
```

DESCRIPTION | The `mllib_ImageMul_Fp_Inp()` function computes the multiplication of two floating-point images on a pixel-by-pixel basis, in place.

It uses the following equation:

$$\text{src1dst}[x][y][i] = \text{src1dst}[x][y][i] * \text{src2}[x][y][i]$$

PARAMETERS | The function takes the following arguments:

src1dst Pointer to source and destination image.

src2 Pointer to second source image.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_ImageMul_Fp\(3MLIB\)](#), [attributes\(5\)](#)

NAME	mllib_ImageMulShift – multiplication						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageMulShift(mllib_image *dst, const mllib_image *src1, const mllib_image *src2, mllib_s32 shift);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageMulShift()</code> function multiplies the pixel values of the two source images. It scales the result by a right shift and writes the result to the destination image on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $dst[x][y][i] = src1[x][y][i] * src2[x][y][i] * 2^{**(-shift)}$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src1</i> Pointer to first source image.</p> <p><i>src2</i> Pointer to second source image.</p> <p><i>shift</i> Right shifting factor. $0 \leq shift \leq 31$.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageMulShift_Inp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageMulShift_Inp(3MLIB)

NAME	mllib_ImageMulShift_Inp – multiplication, in place						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageMulShift_Inp(mllib_image *<i>src1dst</i>, const mllib_image *<i>src2</i>, mllib_s32 <i>shift</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageMulShift_Inp()</code> function multiplies the pixel values of the two source images in place. It scales the result by a right shift and writes the result to the destination image on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $\text{src1dst}[x][y][i] = \text{src1dst}[x][y][i] * \text{src2}[x][y][i] * 2^{**(-\text{shift})}$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>src1dst</i> Pointer to source and destination image.</p> <p><i>src2</i> Pointer to second source image.</p> <p><i>shift</i> Right shifting factor. $0 \leq \text{shift} \leq 31$.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageMulShift(3MLIB)</code> , <code>attributes(5)</code>						

- NAME** mlib_ImageNot – Not
- SYNOPSIS**

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_ImageNot(mlib_image *dst, const mlib_image *src);
```
- DESCRIPTION** The `mlib_ImageNot()` function computes the logical Not of each pixel in the source image.
- It uses the following equation:
- $$dst[x][y][i] = \sim src[x][y][i]$$
- The data type of the images can be `MLIB_BIT`, `MLIB_BYTE`, `MLIB_SHORT`, `MLIB_USHORT`, or `MLIB_INT`.
- PARAMETERS** The function takes the following arguments:
- dst* Pointer to destination image.
- src* Pointer to source image.
- RETURN VALUES** The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.
- ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mlib_ImageNot_Inp(3MLIB)`, `attributes(5)`

mllib_ImageNotAnd(3MLIB)

NAME	mllib_ImageNotAnd – NotAnd						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageNotAnd(mllib_image *<i>dst</i>, const mllib_image *<i>src1</i>, const mllib_image *<i>src2</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageNotAnd()</code> function computes the logical And of the first source image with the second source image and then takes the logical Not of that result on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $\text{dst}[x][y][i] = \sim(\text{src1}[x][y][i] \& \text{src2}[x][y][i])$ <p>The data type of the images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src1</i> Pointer to first source image.</p> <p><i>src2</i> Pointer to second source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageNotAnd_Inp(3MLIB)</code> , <code>attributes(5)</code>						

NAME | mllib_ImageNotAnd_Inp – NotAnd, in place

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageNotAnd_Inp(mllib_image *src1dst, const
mllib_image *src2);
```

DESCRIPTION | The `mllib_ImageNotAnd_Inp()` function computes the logical And of the first source image with the second source images and then takes the logical Not of that result on a pixel-by-pixel basis, in place.

It uses the following equation:

$$src1dst[x][y][i] = \sim(src1dst[x][y][i] \& (src2[x][y][i]))$$

The data type of the images can be `MLIB_BIT`, `MLIB_BYTE`, `MLIB_SHORT`, `MLIB_USHORT`, or `MLIB_INT`.

PARAMETERS | The function takes the following arguments:

src1dst | Pointer to first source and destination image.

src2 | Pointer to second source image.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_ImageNotAnd\(3MLIB\)](#), [attributes\(5\)](#)

mllib_ImageNot_Inp(3MLIB)

NAME	mllib_ImageNot_Inp – Not						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageNot_Inp(mllib_image *srcdst);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageNot_Inp()</code> function computes the logical Not of each pixel in the source image, in place.</p> <p>It uses the following equation:</p> $\text{srcdst}[x][y][i] = \sim\text{srcdst}[x][y][i]$ <p>The data type of the images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>srcdst</i> Pointer to source and destination image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageNot(3MLIB)</code> , <code>attributes(5)</code>						

NAME	mllib_ImageNotOr – NotOr
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmlib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageNotOr(mllib_image *<i>dst</i>, const mllib_image *<i>src1</i>, const mllib_image *<i>src2</i>);</pre>
DESCRIPTION	<p>The <code>mllib_ImageNotOr()</code> function computes the logical Or of the first and second source images and then takes the logical Not of that result on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sim(src1[x][y][i] src2[x][y][i])$ <p>The data type of the images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src1</i> Pointer to first source image.</p> <p><i>src2</i> Pointer to second source image.</p>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_ImageNotOr_Inp\(3MLIB\)](#), [attributes\(5\)](#)

mllib_ImageNotOr_Inp(3MLIB)

NAME | mllib_ImageNotOr_Inp – NotOr, in place

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageNotOr_Inp(mllib_image *src1dst, const
    mllib_image *src2);
```

DESCRIPTION | The `mllib_ImageNotOr_Inp()` function computes the logical Or of the first and second source images and then takes the logical Not of that result on a pixel-by-pixel basis, in place.

It uses the following equation:

$$\text{src1dst}[x][y][i] = \sim(\text{src1dst}[x][y][i] | \text{src2}[x][y][i])$$

The data type of the images can be `MLIB_BIT`, `MLIB_BYTE`, `MLIB_SHORT`, `MLIB_USHORT`, or `MLIB_INT`.

PARAMETERS | The function takes the following arguments:

src1dst | Pointer to first source and destination image.

src2 | Pointer to second source image.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageNotOr(3MLIB)`, `attributes(5)`

NAME	mllib_ImageNotXor – NotXor						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageNotXor(mllib_image *<i>dst</i>, const mllib_image *<i>src1</i>, const mllib_image *<i>src2</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageNotXor()</code> function computes the logical exclusive Or of the first and second source images and then takes the logical Not of that result on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sim(src1[x][y][i] \wedge src2[x][y][i])$ <p>The data type of the images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src1</i> Pointer to first source image.</p> <p><i>src2</i> Pointer to second source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageNotXor_Inp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageNotXor_Inp(3MLIB)

NAME | mllib_ImageNotXor_Inp – NotXor, in place

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageNotXor_Inp(mllib_image *src1dst, const
mllib_image *src2);
```

DESCRIPTION | The `mllib_ImageNotXor_Inp()` function computes the logical exclusive Or of the first and second source images and then takes the logical Not of that result on a pixel-by-pixel basis, in place.

It uses the following equation:

$$\text{src1dst}[x][y][i] = \sim(\text{src1dst}[x][y][i]) \wedge (\text{src2}[x][y][i])$$

The data type of the images can be `MLIB_BIT`, `MLIB_BYTE`, `MLIB_SHORT`, `MLIB_USHORT`, or `MLIB_INT`.

PARAMETERS | The function takes the following arguments:

src1dst | Pointer to first source and destination image.

src2 | Pointer to second source image.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_ImageNotXor\(3MLIB\)](#), `attributes(5)`

NAME	mllib_ImageOr – Or						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageOr(mllib_image *<i>dst</i>, const mllib_image *<i>src1</i>, const mllib_image *<i>src2</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageOr()</code> function computes the logical Or of the first source image with the second source image on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $dst[x][y][i] = src1[x][y][i] src2[x][y][i]$ <p>The data type of the images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src1</i> Pointer to first source image.</p> <p><i>src2</i> Pointer to second source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageOr_Inp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageOr_Inp(3MLIB)

NAME	mllib_ImageOr_Inp – Or, in place						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageOr_Inp(mllib_image *src1dst, const mllib_image *src2);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageOr_Inp()</code> function computes the logical Or of the first source image with the second source image on a pixel-by-pixel basis, in place.</p> <p>It uses the following equation:</p> $\text{src1dst}[x][y][i] = \text{src1dst}[x][y][i] \mid \text{src2}[x][y][i]$ <p>The data type of the images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>src1dst</i> Pointer to first source and destination image.</p> <p><i>src2</i> Pointer to second source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageOr(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageOrNot1_Inp – OrNot, in place

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageOrNot1_Inp(mllib_image *src1dst, const
    mllib_image *src2);
```

DESCRIPTION The `mllib_ImageOrNot1_Inp()` function computes the logical Not of the second source image and then takes the logical Or of that result with the first source image on a pixel-by-pixel basis, in place.

It uses the following equation:

$$\text{src1dst}[x][y][i] = \text{src1dst}[x][y][i] \mid (\sim\text{src2}[x][y][i])$$

The data type of the images can be `MLIB_BIT`, `MLIB_BYTE`, `MLIB_SHORT`, `MLIB_USHORT`, or `MLIB_INT`.

PARAMETERS The function takes the following arguments:

src1dst Pointer to first source and destination image.

src2 Pointer to second source image.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageOrNot(3MLIB)`, `mllib_ImageOrNot2_Inp(3MLIB)`, `attributes(5)`

mllib_ImageOrNot2_Inp(3MLIB)

NAME | mllib_ImageOrNot2_Inp – OrNot, in place

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageOrNot2_Inp(mllib_image *src2dst, const
mllib_image *src1);
```

DESCRIPTION | The `mllib_ImageOrNot2_Inp()` function computes the logical Not of the second source image and then takes the logical Or of that result with the first source image on a pixel-by-pixel basis, in place.

It uses the following equation:

$$\text{src2dst}[x][y][i] = \text{src1}[x][y][i] \mid (\sim\text{src2dst}[x][y][i])$$

The data type of the images can be `MLIB_BIT`, `MLIB_BYTE`, `MLIB_SHORT`, `MLIB_USHORT`, or `MLIB_INT`.

PARAMETERS | The function takes the following arguments:

src2dst | Pointer to second source and destination image.

src1 | Pointer to first source image.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageOrNot(3MLIB)`, `mllib_ImageOrNot1_Inp(3MLIB)`, `attributes(5)`

NAME	mlib_ImageOrNot – OrNot						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmlib [<i>library...</i>] #include <mlib.h> mlib_status mlib_ImageOrNot(mlib_image *<i>dst</i>, const mlib_image *<i>src1</i>, const mlib_image *<i>src2</i>);</pre>						
DESCRIPTION	<p>The <code>mlib_ImageOrNot()</code> function computes the logical Not of the second source image and then takes the logical Or of that result with the first source image on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $dst[x][y][i] = src1[x][y][i] (\sim src2[x][y][i])$ <p>The data type of the images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src1</i> Pointer to first source image.</p> <p><i>src2</i> Pointer to second source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mlib_ImageOrNot1_Inp(3MLIB)</code> , <code>mlib_ImageOrNot2_Inp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImagePolynomialWarp(3MLIB)

NAME	mllib_ImagePolynomialWarp – polynomial-based image warp
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImagePolynomialWarp(mllib_image *dst, const mllib_image *src, const mllib_d64 *xCoeffs, const mllib_d64 *yCoeffs, mllib_s32 n, mllib_d64 preShiftX, mllib_d64 preShiftY, mllib_d64 postShiftX, mllib_d64 postShiftY, mllib_d64 preScaleX, mllib_d64 preScaleY, mllib_d64 postScaleX, mllib_d64 postScaleY, mllib_filter filter, mllib_edge edge);</pre>
DESCRIPTION	<p>The <code>mllib_ImagePolynomialWarp()</code> function performs a polynomial-based image warp.</p> <p>The images must have the same type, and the same number of channels. The images can have 1, 2, 3, or 4 channels. The data type of the images can be <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>. The source and destination images may have different sizes.</p> <p>The <code>xCoeffs</code> and <code>yCoeffs</code> parameters must contain the same number of coefficients of the form $(n + 1)(n + 2)/2$ for some n, where n is the degree power of the polynomial. The coefficients, in order, are associated with the terms:</p> <pre>1, x, y, x**2, x*y, y**2, ..., x**n, x**(n-1)*y, ..., x*y**(n-1), y**n</pre> <p>and coefficients of value 0 cannot be omitted.</p> <p>The image pixels are assumed to be centered at .5 coordinate points. In other words, the upper-left corner pixel of an image is located at (0.5, 0.5).</p> <p>For each pixel in the destination image, its center point D is backward mapped to a point S in the source image. Then the source pixels with their centers surrounding point S are selected to do one of the interpolations specified by the <i>filter</i> parameter to generate the pixel value for point D.</p> <p>The mapping is defined by the two bivariate polynomial functions $X(x, y)$ and $Y(x, y)$ that map destination (x, y) coordinates to source X and Y positions respectively.</p> <p>The functions $X(x, y)$ and $Y(x, y)$ are:</p> <pre>preX = (x + preShiftX)*preScaleX preY = (y + preShiftY)*preScaleY warpedX = SUM_{i=0}^n {SUM_{j=0}^i {xCoeffs_ij * preX**(i-j) * preY**j}} warpedY = SUM_{i=0}^n {SUM_{j=0}^i {yCoeffs_ij * preX**(i-j) * preY**j}}</pre>

mllib_ImagePolynomialWarp(3MLIB)

$$X(x, y) = \text{warpedX} * \text{postScaleX} - \text{postShiftX}$$
$$Y(x, y) = \text{warpedY} * \text{postScaleY} - \text{postShiftY}$$

The destination (x, y) coordinates are pre-shifted by $(\text{preShiftX}, \text{preShiftY})$ and pre-scaled by the factors preScaleX and preScaleY prior to the evaluation of the polynomial. The results of the polynomial evaluations are scaled by postScaleX and postScaleY , and then shifted by $(-\text{postShiftX}, -\text{postShiftY})$ to produce the source pixel coordinates. This process allows for better precision of the results and supports tiled images.

PARAMETERS

The function takes the following arguments:

<i>dst</i>	Pointer to destination image.
<i>src</i>	Pointer to source image.
<i>xCoeffs</i>	Destination to source transform coefficients for the X coordinate.
<i>yCoeffs</i>	Destination to source transform coefficients for the Y coordinate.
<i>n</i>	Degree power of the polynomial.
<i>preShiftX</i>	Displacement to apply to destination X positions.
<i>preShiftY</i>	Displacement to apply to destination Y positions.
<i>postShiftX</i>	Displacement to apply to source X positions.
<i>postShiftY</i>	Displacement to apply to source Y positions.
<i>preScaleX</i>	Scale factor to apply to destination X positions.
<i>preScaleY</i>	Scale factor to apply to destination Y positions.
<i>postScaleX</i>	Scale factor to apply to source X positions.
<i>postScaleY</i>	Scale factor to apply to source Y positions.
<i>filter</i>	Type of resampling filter. It can be one of the following: MLIB_NEAREST MLIB_BILINEAR MLIB_BICUBIC MLIB_BICUBIC2
<i>edge</i>	Type of edge condition. It can be one of the following: MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_SRC_PADDED

RETURN VALUES

The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

mllib_ImagePolynomialWarp(3MLIB)

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImagePolynomialWarp_Fp(3MLIB)`,
`mllib_ImagePolynomialWarpTable(3MLIB)`,
`mllib_ImagePolynomialWarpTable_Fp(3MLIB)`, `attributes(5)`

NAME	mllib_ImagePolynomialWarp_Fp – polynomial-based image warp
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImagePolynomialWarp_Fp(mllib_image *dst, const mllib_image *src, const mllib_d64 *xCoeffs, const mllib_d64 *yCoeffs, mllib_s32 n, mllib_d64 preShiftX, mllib_d64 preShiftY, mllib_d64 postShiftX, mllib_d64 postShiftY, mllib_d64 preScaleX, mllib_d64 preScaleY, mllib_d64 postScaleX, mllib_d64 postScaleY, mllib_filter filter, mllib_edge edge);</pre>
DESCRIPTION	<p>The <code>mllib_ImagePolynomialWarp_Fp()</code> function performs a polynomial-based image warp on a floating-point image.</p> <p>The images must have the same type, and the same number of channels. The images can have 1, 2, 3, or 4 channels. The data type of the images can be <code>MLIB_FLOAT</code> or <code>MLIB_DOUBLE</code>. The source and destination images may have different sizes.</p> <p>The <code>xCoeffs</code> and <code>yCoeffs</code> parameters must contain the same number of coefficients of the form $(n + 1)(n + 2)/2$ for some n, where n is the degree power of the polynomial. The coefficients, in order, are associated with the terms:</p> <pre>1, x, y, x**2, x*y, y**2, ..., x**n, x**(n-1)*y, ..., x*y**(n-1), y**n</pre> <p>and coefficients of value 0 cannot be omitted.</p> <p>The image pixels are assumed to be centered at .5 coordinate points. In other words, the upper-left corner pixel of an image is located at (0.5, 0.5).</p> <p>For each pixel in the destination image, its center point D is backward mapped to a point S in the source image. Then the source pixels with their centers surrounding point S are selected to do one of the interpolations specified by the <code>filter</code> parameter to generate the pixel value for point D.</p> <p>The mapping is defined by the two bivariate polynomial functions $X(x, y)$ and $Y(x, y)$ that map destination (x, y) coordinates to source X and Y positions respectively.</p> <p>The functions $X(x, y)$ and $Y(x, y)$ are:</p> <pre>preX = (x + preShiftX)*preScaleX preY = (y + preShiftY)*preScaleY warpedX = SUM_{i=0}^n {SUM_{j=0}^i {xCoeffs_ij * preX**(i-j) * preY**j}} warpedY = SUM_{i=0}^n {SUM_{j=0}^i {yCoeffs_ij * preX**(i-j) * preY**j}} X(x, y) = warpedX*postScaleX - postShiftX</pre>

mllib_ImagePolynomialWarp_Fp(3MLIB)

$Y(x, y) = \text{warpedY} * \text{postScaleY} - \text{postShiftY}$

The destination (x, y) coordinates are pre-shifted by $(\text{preShiftX}, \text{preShiftY})$ and pre-scaled by the factors preScaleX and preScaleY prior to the evaluation of the polynomial. The results of the polynomial evaluations are scaled by postScaleX and postScaleY , and then shifted by $(-\text{postShiftX}, -\text{postShiftY})$ to produce the source pixel coordinates. This process allows for better precision of the results and supports tiled images.

PARAMETERS

The function takes the following arguments:

<i>dst</i>	Pointer to destination image.
<i>src</i>	Pointer to source image.
<i>xCoeffs</i>	Destination to source transform coefficients for the X coordinate.
<i>yCoeffs</i>	Destination to source transform coefficients for the Y coordinate.
<i>n</i>	Degree power of the polynomial.
<i>preShiftX</i>	Displacement to apply to destination X positions.
<i>preShiftY</i>	Displacement to apply to destination Y positions.
<i>postShiftX</i>	Displacement to apply to source X positions.
<i>postShiftY</i>	Displacement to apply to source Y positions.
<i>preScaleX</i>	Scale factor to apply to destination X positions.
<i>preScaleY</i>	Scale factor to apply to destination Y positions.
<i>postScaleX</i>	Scale factor to apply to source X positions.
<i>postScaleY</i>	Scale factor to apply to source Y positions.
<i>filter</i>	Type of resampling filter. It can be one of the following: MLIB_NEAREST MLIB_BILINEAR MLIB_BICUBIC MLIB_BICUBIC2
<i>edge</i>	Type of edge condition. It can be one of the following: MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_SRC_PADDED

RETURN VALUES

The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

mllib_ImagePolynomialWarp_Fp(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

`mllib_ImagePolynomialWarp(3MLIB)`,
`mllib_ImagePolynomialWarpTable(3MLIB)`,
`mllib_ImagePolynomialWarpTable_Fp(3MLIB)`, `attributes(5)`

mllib_ImagePolynomialWarpTable(3MLIB)

NAME	mllib_ImagePolynomialWarpTable – polynomial-based image warp with table-driven interpolation
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImagePolynomialWarpTable(mllib_image *dst, const mllib_image *src, const mllib_d64 *xCoeffs, const mllib_d64 *yCoeffs, mllib_s32 n, mllib_d64 preShiftX, mllib_d64 preShiftY, mllib_d64 postShiftX, mllib_d64 postShiftY, mllib_d64 preScaleX, mllib_d64 preScaleY, mllib_d64 postScaleX, mllib_d64 postScaleY, const void *interp_table, mllib_edge edge);</pre>
DESCRIPTION	<p>The <code>mllib_ImagePolynomialWarpTable()</code> function performs a polynomial-based image warp with table-driven interpolation.</p> <p>The images must have the same type, and the same number of channels. The images can have 1, 2, 3, or 4 channels. The data type of the images can be <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>. The source and destination images may have different sizes.</p> <p>The <code>xCoeffs</code> and <code>yCoeffs</code> parameters must contain the same number of coefficients of the form $(n + 1)(n + 2)/2$ for some n, where n is the degree power of the polynomial. The coefficients, in order, are associated with the terms:</p> <pre>1, x, y, x**2, x*y, y**2, ..., x**n, x**(n-1)*y, ..., x*y**(n-1), y**n</pre> <p>and coefficients of value 0 cannot be omitted.</p> <p>The image pixels are assumed to be centered at .5 coordinate points. In other words, the upper-left corner pixel of an image is located at (0.5, 0.5).</p> <p>For each pixel in the destination image, its center point D is backward mapped to a point S in the source image. Then the source pixels with their centers surrounding point S are selected to do the interpolation specified by <code>interp_table</code> to generate the pixel value for point D.</p> <p>The mapping is defined by the two bivariate polynomial functions $X(x, y)$ and $Y(x, y)$ that map destination (x, y) coordinates to source X and Y positions respectively.</p> <p>The functions $X(x, y)$ and $Y(x, y)$ are:</p> <pre>preX = (x + preShiftX)*preScaleX preY = (y + preShiftY)*preScaleY warpedX = SUM_{i=0}^n {SUM_{j=0}^i {xCoeffs_ij * preX**(i-j) * preY**j}} warpedY = SUM_{i=0}^n {SUM_{j=0}^i {yCoeffs_ij * preX**(i-j) * preY**j}}</pre>

mllib_ImagePolynomialWarpTable(3MLIB)

$X(x, y) = \text{warpedX} * \text{postScaleX} - \text{postShiftX}$

$Y(x, y) = \text{warpedY} * \text{postScaleY} - \text{postShiftY}$

The destination (x, y) coordinates are pre-shifted by $(\text{preShiftX}, \text{preShiftY})$ and pre-scaled by the factors preScaleX and preScaleY prior to the evaluation of the polynomial. The results of the polynomial evaluations are scaled by postScaleX and postScaleY , and then shifted by $(-\text{postShiftX}, -\text{postShiftY})$ to produce the source pixel coordinates. This process allows for better precision of the results and supports tiled images.

PARAMETERS

The function takes the following arguments:

<i>dst</i>	Pointer to destination image.
<i>src</i>	Pointer to source image.
<i>xCoeffs</i>	Destination to source transform coefficients for the X coordinate.
<i>yCoeffs</i>	Destination to source transform coefficients for the Y coordinate.
<i>n</i>	Degree power of the polynomial.
<i>preShiftX</i>	Displacement to apply to destination X positions.
<i>preShiftY</i>	Displacement to apply to destination Y positions.
<i>postShiftX</i>	Displacement to apply to source X positions.
<i>postShiftY</i>	Displacement to apply to source Y positions.
<i>preScaleX</i>	Scale factor to apply to destination X positions.
<i>preScaleY</i>	Scale factor to apply to destination Y positions.
<i>postScaleX</i>	Scale factor to apply to source X positions.
<i>postScaleY</i>	Scale factor to apply to source Y positions.
<i>interp_table</i>	Pointer to an interpolation table. The table is created by the <code>mllib_ImageInterpTableCreate()</code> function.
<i>edge</i>	Type of edge condition. It can be one of the following: MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_SRC_PADDED

RETURN VALUES

The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

mllib_ImagePolynomialWarpTable(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	MT-Safe

SEE ALSO `mllib_ImageInterpTableCreate(3MLIB)`,
`mllib_ImageInterpTableDelete(3MLIB)`,
`mllib_ImagePolynomialWarpTable_Fp(3MLIB)`,
`mllib_ImagePolynomialWarp(3MLIB)`,
`mllib_ImagePolynomialWarp_Fp(3MLIB)`, `attributes(5)`

NAME	mllib_ImagePolynomialWarpTable_Fp – polynomial-based image warp with table-driven interpolation
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImagePolynomialWarpTable_Fp(mllib_image *dst, const mllib_image *src, const mllib_d64 *xCoeffs, const mllib_d64 *yCoeffs, mllib_s32 n, mllib_d64 preShiftX, mllib_d64 preShiftY, mllib_d64 postShiftX, mllib_d64 postShiftY, mllib_d64 preScaleX, mllib_d64 preScaleY, mllib_d64 postScaleX, mllib_d64 postScaleY, const void *interp_table, mllib_edge edge);</pre>
DESCRIPTION	<p>The <code>mllib_ImagePolynomialWarpTable_Fp()</code> function performs a polynomial-based image warp on a floating-point image with table-driven interpolation.</p> <p>The images must have the same type, and the same number of channels. The images can have 1, 2, 3, or 4 channels. The data type of the images can be <code>MLIB_FLOAT</code> or <code>MLIB_DOUBLE</code>. The source and destination images may have different sizes.</p> <p>The <code>xCoeffs</code> and <code>yCoeffs</code> parameters must contain the same number of coefficients of the form $(n + 1)(n + 2)/2$ for some n, where n is the degree power of the polynomial. The coefficients, in order, are associated with the terms:</p> <pre>1, x, y, x**2, x*y, y**2, ..., x**n, x**(n-1)*y, ..., x*y**(n-1), y**n</pre> <p>and coefficients of value 0 cannot be omitted.</p> <p>The image pixels are assumed to be centered at .5 coordinate points. In other words, the upper-left corner pixel of an image is located at (0.5, 0.5).</p> <p>For each pixel in the destination image, its center point D is backward mapped to a point S in the source image. Then the source pixels with their centers surrounding point S are selected to do the interpolation specified by <code>interp_table</code> to generate the pixel value for point D.</p> <p>The mapping is defined by the two bivariate polynomial functions $X(x, y)$ and $Y(x, y)$ that map destination (x, y) coordinates to source X and Y positions respectively.</p> <p>The functions $X(x, y)$ and $Y(x, y)$ are:</p> <pre>preX = (x + preShiftX)*preScaleX preY = (y + preShiftY)*preScaleY</pre> $\text{warpedX} = \sum_{i=0}^n \left\{ \sum_{j=0}^i \{xCoeffs_{ij} * preX^{i-j} * preY^{j}\} \right\}$ $\text{warpedY} = \sum_{i=0}^n \left\{ \sum_{j=0}^i \{yCoeffs_{ij} * preX^{i-j} * preY^{j}\} \right\}$

mllib_ImagePolynomialWarpTable_Fp(3MLIB)

$$X(x, y) = \text{warpedX} * \text{postScaleX} - \text{postShiftX}$$

$$Y(x, y) = \text{warpedY} * \text{postScaleY} - \text{postShiftY}$$

The destination (x, y) coordinates are pre-shifted by $(\text{preShiftX}, \text{preShiftY})$ and pre-scaled by the factors preScaleX and preScaleY prior to the evaluation of the polynomial. The results of the polynomial evaluations are scaled by postScaleX and postScaleY , and then shifted by $(-\text{postShiftX}, -\text{postShiftY})$ to produce the source pixel coordinates. This process allows for better precision of the results and supports tiled images.

PARAMETERS The function takes the following arguments:

<i>dst</i>	Pointer to destination image.
<i>src</i>	Pointer to source image.
<i>xCoeffs</i>	Destination to source transform coefficients for the X coordinate.
<i>yCoeffs</i>	Destination to source transform coefficients for the Y coordinate.
<i>n</i>	Degree power of the polynomial.
<i>preShiftX</i>	Displacement to apply to destination X positions.
<i>preShiftY</i>	Displacement to apply to destination Y positions.
<i>postShiftX</i>	Displacement to apply to source X positions.
<i>postShiftY</i>	Displacement to apply to source Y positions.
<i>preScaleX</i>	Scale factor to apply to destination X positions.
<i>preScaleY</i>	Scale factor to apply to destination Y positions.
<i>postScaleX</i>	Scale factor to apply to source X positions.
<i>postScaleY</i>	Scale factor to apply to source Y positions.
<i>interp_table</i>	Pointer to an interpolation table. The table is created by the <code>mllib_ImageInterpTableCreate()</code> function.
<i>edge</i>	Type of edge condition. It can be one of the following: MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_SRC_PADDED

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

mllib_ImagePolynomialWarpTable_Fp(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	MT-Safe

SEE ALSO

`mllib_ImageInterpTableCreate(3MLIB)`,
`mllib_ImageInterpTableDelete(3MLIB)`,
`mllib_ImagePolynomialWarpTable(3MLIB)`,
`mllib_ImagePolynomialWarp(3MLIB)`,
`mllib_ImagePolynomialWarp_Fp(3MLIB)`, `attributes(5)`

mllib_ImageRankFilter3x3(3MLIB)

NAME	mllib_ImageRankFilter3x3 – 3x3 rank filter						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageRankFilter3x3(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, mllib_s32 <i>rank</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageRankFilter3x3()</code> function performs 3x3 rank filtering on an image. Each pixel of the destination image is the pixel with the user-specified rank in the filter window.</p> <p>The source and destination images must be single-channel images.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>dst</i></td><td>Pointer to destination image.</td></tr><tr><td><i>src</i></td><td>Pointer to source image.</td></tr><tr><td><i>rank</i></td><td>The rank of the destination pixel. The pixel with minimum value is designated rank 0.</td></tr></table>	<i>dst</i>	Pointer to destination image.	<i>src</i>	Pointer to source image.	<i>rank</i>	The rank of the destination pixel. The pixel with minimum value is designated rank 0.
<i>dst</i>	Pointer to destination image.						
<i>src</i>	Pointer to source image.						
<i>rank</i>	The rank of the destination pixel. The pixel with minimum value is designated rank 0.						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	<p>See <code>attributes(5)</code> for descriptions of the following attributes:</p> <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<p><code>mllib_ImageMaxFilter3x3(3MLIB)</code>, <code>mllib_ImageMaxFilter3x3_Fp(3MLIB)</code>, <code>mllib_ImageMaxFilter5x5(3MLIB)</code>, <code>mllib_ImageMaxFilter5x5_Fp(3MLIB)</code>, <code>mllib_ImageMaxFilter7x7(3MLIB)</code>, <code>mllib_ImageMaxFilter7x7_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3_US(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5_US(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7_US(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN_US(3MLIB)</code>, <code>mllib_ImageMinFilter3x3(3MLIB)</code>, <code>mllib_ImageMinFilter3x3_Fp(3MLIB)</code>, <code>mllib_ImageMinFilter5x5(3MLIB)</code>, <code>mllib_ImageMinFilter5x5_Fp(3MLIB)</code>, <code>mllib_ImageMinFilter7x7(3MLIB)</code>, <code>mllib_ImageMinFilter7x7_Fp(3MLIB)</code>,</p>						

mllib_ImageRankFilter3x3(3MLIB)

```
mllib_ImageRankFilter3x3_Fp(3MLIB),  
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),  
mllib_ImageRankFilter5x5_Fp(3MLIB),  
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),  
mllib_ImageRankFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),  
mllib_ImageRankFilterMxN_Fp(3MLIB),  
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

mllib_ImageRankFilter3x3_Fp(3MLIB)

NAME	mllib_ImageRankFilter3x3_Fp – 3x3 rank filter						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageRankFilter3x3_Fp(mllib_image *dst, const mllib_image *src, mllib_s32 rank);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageRankFilter3x3_Fp()</code> function performs floating-point 3x3 rank filtering on an image. Each pixel of the destination image is the pixel with the user-specified rank in the filter window.</p> <p>The source and destination images must be single-channel images.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>dst</i></td><td>Pointer to destination image.</td></tr><tr><td><i>src</i></td><td>Pointer to source image.</td></tr><tr><td><i>rank</i></td><td>The rank of the destination pixel. The pixel with minimum value is designated rank 0.</td></tr></table>	<i>dst</i>	Pointer to destination image.	<i>src</i>	Pointer to source image.	<i>rank</i>	The rank of the destination pixel. The pixel with minimum value is designated rank 0.
<i>dst</i>	Pointer to destination image.						
<i>src</i>	Pointer to source image.						
<i>rank</i>	The rank of the destination pixel. The pixel with minimum value is designated rank 0.						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageMaxFilter3x3(3MLIB)</code> , <code>mllib_ImageMaxFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMaxFilter5x5(3MLIB)</code> , <code>mllib_ImageMaxFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMaxFilter7x7(3MLIB)</code> , <code>mllib_ImageMaxFilter7x7_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3_US(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5_US(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7_US(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN_US(3MLIB)</code> , <code>mllib_ImageMinFilter3x3(3MLIB)</code> , <code>mllib_ImageMinFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMinFilter5x5(3MLIB)</code> , <code>mllib_ImageMinFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMinFilter7x7(3MLIB)</code> , <code>mllib_ImageMinFilter7x7_Fp(3MLIB)</code> , <code>mllib_ImageRankFilter3x3(3MLIB)</code> ,						

`mllib_ImageRankFilter3x3_Fp(3MLIB)`

```
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),  
mllib_ImageRankFilter5x5_Fp(3MLIB),  
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),  
mllib_ImageRankFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),  
mllib_ImageRankFilterMxN_Fp(3MLIB),  
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

mllib_ImageRankFilter3x3_US(3MLIB)

NAME	mllib_ImageRankFilter3x3_US – 3x3 rank filter, unsigned								
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageRankFilter3x3_US(mllib_image *dst, const mllib_image *src, mllib_s32 rank, mllib_s32 bits);</pre>								
DESCRIPTION	<p>The <code>mllib_ImageRankFilter3x3_US()</code> function performs 3x3 rank filtering on an <code>MLIB_SHORT</code> type of image that contains unsigned data. Each pixel of the destination image is the pixel with the user-specified rank in the filter window.</p> <p>The source and destination images must be single-channel images.</p>								
PARAMETERS	<p>The function takes the following arguments:</p> <table border="0"> <tr> <td style="padding-right: 20px;"><i>dst</i></td> <td>Pointer to destination image.</td> </tr> <tr> <td><i>src</i></td> <td>Pointer to source image.</td> </tr> <tr> <td><i>rank</i></td> <td>The rank of the destination pixel. The pixel with minimum value is designated rank 0.</td> </tr> <tr> <td><i>bits</i></td> <td>The number of unsigned bits for pixel dynamic range. $9 \leq \text{bits} \leq 15$.</td> </tr> </table>	<i>dst</i>	Pointer to destination image.	<i>src</i>	Pointer to source image.	<i>rank</i>	The rank of the destination pixel. The pixel with minimum value is designated rank 0.	<i>bits</i>	The number of unsigned bits for pixel dynamic range. $9 \leq \text{bits} \leq 15$.
<i>dst</i>	Pointer to destination image.								
<i>src</i>	Pointer to source image.								
<i>rank</i>	The rank of the destination pixel. The pixel with minimum value is designated rank 0.								
<i>bits</i>	The number of unsigned bits for pixel dynamic range. $9 \leq \text{bits} \leq 15$.								
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .								
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:								
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe		
ATTRIBUTE TYPE	ATTRIBUTE VALUE								
Interface Stability	Evolving								
MT-Level	MT-Safe								
SEE ALSO	<p><code>mllib_ImageMaxFilter3x3(3MLIB)</code>, <code>mllib_ImageMaxFilter3x3_Fp(3MLIB)</code>, <code>mllib_ImageMaxFilter5x5(3MLIB)</code>, <code>mllib_ImageMaxFilter5x5_Fp(3MLIB)</code>, <code>mllib_ImageMaxFilter7x7(3MLIB)</code>, <code>mllib_ImageMaxFilter7x7_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3_US(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5_US(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7_US(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN_US(3MLIB)</code>, <code>mllib_ImageMinFilter3x3(3MLIB)</code>,</p>								

mllib_ImageRankFilter3x3_US(3MLIB)

```
mllib_ImageMinFilter3x3_Fp(3MLIB), mllib_ImageMinFilter5x5(3MLIB),  
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7(3MLIB),  
mllib_ImageMinFilter7x7_Fp(3MLIB), mllib_ImageRankFilter3x3(3MLIB),  
mllib_ImageRankFilter3x3_Fp(3MLIB), mllib_ImageRankFilter5x5(3MLIB),  
mllib_ImageRankFilter5x5_Fp(3MLIB),  
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),  
mllib_ImageRankFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),  
mllib_ImageRankFilterMxN_Fp(3MLIB),  
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

mllib_ImageRankFilter5x5(3MLIB)

NAME	mllib_ImageRankFilter5x5 – 5x5 rank filter						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageRankFilter5x5(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, mllib_s32 <i>rank</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageRankFilter5x5()</code> function performs 5x5 rank filtering on an image. Each pixel of the destination image is the pixel with the user-specified rank in the filter window.</p> <p>The source and destination images must be single-channel images.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>dst</i></td><td>Pointer to destination image.</td></tr><tr><td><i>src</i></td><td>Pointer to source image.</td></tr><tr><td><i>rank</i></td><td>The rank of the destination pixel. The pixel with minimum value is designated rank 0.</td></tr></table>	<i>dst</i>	Pointer to destination image.	<i>src</i>	Pointer to source image.	<i>rank</i>	The rank of the destination pixel. The pixel with minimum value is designated rank 0.
<i>dst</i>	Pointer to destination image.						
<i>src</i>	Pointer to source image.						
<i>rank</i>	The rank of the destination pixel. The pixel with minimum value is designated rank 0.						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageMaxFilter3x3(3MLIB)</code> , <code>mllib_ImageMaxFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMaxFilter5x5(3MLIB)</code> , <code>mllib_ImageMaxFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMaxFilter7x7(3MLIB)</code> , <code>mllib_ImageMaxFilter7x7_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3_US(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5_US(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7_US(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN_US(3MLIB)</code> , <code>mllib_ImageMinFilter3x3(3MLIB)</code> , <code>mllib_ImageMinFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMinFilter5x5(3MLIB)</code> , <code>mllib_ImageMinFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMinFilter7x7(3MLIB)</code> , <code>mllib_ImageMinFilter7x7_Fp(3MLIB)</code> , <code>mllib_ImageRankFilter3x3(3MLIB)</code> ,						

mllib_ImageRankFilter5x5(3MLIB)

```
mllib_ImageRankFilter3x3_Fp(3MLIB),  
mllib_ImageRankFilter3x3_US(3MLIB),  
mllib_ImageRankFilter5x5_Fp(3MLIB),  
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),  
mllib_ImageRankFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),  
mllib_ImageRankFilterMxN_Fp(3MLIB),  
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

mllib_ImageRankFilter5x5_Fp(3MLIB)

NAME	mllib_ImageRankFilter5x5_Fp – 5x5 rank filter						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageRankFilter5x5_Fp(mllib_image *dst, const mllib_image *src, mllib_s32 rank);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageRankFilter5x5_Fp()</code> function performs floating-point 5x5 rank filtering on an image. Each pixel of the destination image is the pixel with the user-specified rank in the filter window.</p> <p>The source and destination images must be single-channel images.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>dst</i></td><td>Pointer to destination image.</td></tr><tr><td><i>src</i></td><td>Pointer to source image.</td></tr><tr><td><i>rank</i></td><td>The rank of the destination pixel. The pixel with minimum value is designated rank 0.</td></tr></table>	<i>dst</i>	Pointer to destination image.	<i>src</i>	Pointer to source image.	<i>rank</i>	The rank of the destination pixel. The pixel with minimum value is designated rank 0.
<i>dst</i>	Pointer to destination image.						
<i>src</i>	Pointer to source image.						
<i>rank</i>	The rank of the destination pixel. The pixel with minimum value is designated rank 0.						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageMaxFilter3x3(3MLIB)</code> , <code>mllib_ImageMaxFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMaxFilter5x5(3MLIB)</code> , <code>mllib_ImageMaxFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMaxFilter7x7(3MLIB)</code> , <code>mllib_ImageMaxFilter7x7_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3_US(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5_US(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7_US(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN_US(3MLIB)</code> , <code>mllib_ImageMinFilter3x3(3MLIB)</code> , <code>mllib_ImageMinFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMinFilter5x5(3MLIB)</code> , <code>mllib_ImageMinFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMinFilter7x7(3MLIB)</code> , <code>mllib_ImageMinFilter7x7_Fp(3MLIB)</code> , <code>mllib_ImageRankFilter3x3(3MLIB)</code> ,						

mllib_ImageRankFilter5x5_Fp(3MLIB)

```
mllib_ImageRankFilter3x3_Fp(3MLIB),  
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),  
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),  
mllib_ImageRankFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),  
mllib_ImageRankFilterMxN_Fp(3MLIB),  
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

mllib_ImageRankFilter5x5_US(3MLIB)

NAME | mllib_ImageRankFilter5x5_US – 5x5 rank filter, unsigned

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageRankFilter5x5_US(mllib_image *dst, const
mllib_image *src, mllib_s32 rank, mllib_s32 bits);
```

DESCRIPTION | The `mllib_ImageRankFilter5x5_US()` function performs 5x5 rank filtering on an `MLIB_SHORT` type of image that contains unsigned data. Each pixel of the destination image is the pixel with the user-specified rank in the filter window.

The source and destination images must be single-channel images.

PARAMETERS | The function takes the following arguments:

<i>dst</i>	Pointer to destination image.
<i>src</i>	Pointer to source image.
<i>rank</i>	The rank of the destination pixel. The pixel with minimum value is designated rank 0.
<i>bits</i>	The number of unsigned bits for pixel dynamic range. $9 \leq \text{bits} \leq 15$.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageMaxFilter3x3(3MLIB)`, `mllib_ImageMaxFilter3x3_Fp(3MLIB)`, `mllib_ImageMaxFilter5x5(3MLIB)`, `mllib_ImageMaxFilter5x5_Fp(3MLIB)`, `mllib_ImageMaxFilter7x7(3MLIB)`, `mllib_ImageMaxFilter7x7_Fp(3MLIB)`, `mllib_ImageMedianFilter3x3(3MLIB)`, `mllib_ImageMedianFilter3x3_Fp(3MLIB)`, `mllib_ImageMedianFilter3x3_US(3MLIB)`, `mllib_ImageMedianFilter5x5(3MLIB)`, `mllib_ImageMedianFilter5x5_Fp(3MLIB)`, `mllib_ImageMedianFilter5x5_US(3MLIB)`, `mllib_ImageMedianFilter7x7(3MLIB)`, `mllib_ImageMedianFilter7x7_Fp(3MLIB)`, `mllib_ImageMedianFilter7x7_US(3MLIB)`, `mllib_ImageMedianFilterMxN(3MLIB)`, `mllib_ImageMedianFilterMxN_Fp(3MLIB)`, `mllib_ImageMedianFilterMxN_US(3MLIB)`, `mllib_ImageMinFilter3x3(3MLIB)`,

mllib_ImageRankFilter5x5_US(3MLIB)

```
mllib_ImageMinFilter3x3_Fp(3MLIB), mllib_ImageMinFilter5x5(3MLIB),  
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7(3MLIB),  
mllib_ImageMinFilter7x7_Fp(3MLIB), mllib_ImageRankFilter3x3(3MLIB),  
mllib_ImageRankFilter3x3_Fp(3MLIB),  
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),  
mllib_ImageRankFilter5x5_Fp(3MLIB), mllib_ImageRankFilter7x7(3MLIB),  
mllib_ImageRankFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),  
mllib_ImageRankFilterMxN_Fp(3MLIB),  
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

mllib_ImageRankFilter7x7(3MLIB)

NAME	mllib_ImageRankFilter7x7 – 7x7 rank filter						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageRankFilter7x7(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, mllib_s32 <i>rank</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageRankFilter7x7()</code> function performs 7x7 rank filtering on an image. Each pixel of the destination image is the pixel with the user-specified rank in the filter window.</p> <p>The source and destination images must be single-channel images.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>dst</i></td><td>Pointer to destination image.</td></tr><tr><td><i>src</i></td><td>Pointer to source image.</td></tr><tr><td><i>rank</i></td><td>The rank of the destination pixel. The pixel with minimum value is designated rank 0.</td></tr></table>	<i>dst</i>	Pointer to destination image.	<i>src</i>	Pointer to source image.	<i>rank</i>	The rank of the destination pixel. The pixel with minimum value is designated rank 0.
<i>dst</i>	Pointer to destination image.						
<i>src</i>	Pointer to source image.						
<i>rank</i>	The rank of the destination pixel. The pixel with minimum value is designated rank 0.						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageMaxFilter3x3(3MLIB)</code> , <code>mllib_ImageMaxFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMaxFilter5x5(3MLIB)</code> , <code>mllib_ImageMaxFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMaxFilter7x7(3MLIB)</code> , <code>mllib_ImageMaxFilter7x7_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3_US(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5_US(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7_US(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN_US(3MLIB)</code> , <code>mllib_ImageMinFilter3x3(3MLIB)</code> , <code>mllib_ImageMinFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMinFilter5x5(3MLIB)</code> , <code>mllib_ImageMinFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMinFilter7x7(3MLIB)</code> , <code>mllib_ImageMinFilter7x7_Fp(3MLIB)</code> , <code>mllib_ImageRankFilter3x3(3MLIB)</code> ,						

mllib_ImageRankFilter7x7(3MLIB)

```
mllib_ImageRankFilter3x3_Fp(3MLIB),  
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),  
mllib_ImageRankFilter5x5_Fp(3MLIB),  
mllib_ImageRankFilter5x5_US(3MLIB),  
mllib_ImageRankFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),  
mllib_ImageRankFilterMxN_Fp(3MLIB),  
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

mllib_ImageRankFilter7x7_Fp(3MLIB)

NAME	mllib_ImageRankFilter7x7_Fp – 7x7 rank filter						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageRankFilter7x7_Fp(mllib_image *dst, const mllib_image *src, mllib_s32 rank);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageRankFilter7x7_Fp()</code> function performs floating-point 7x7 rank filtering on an image. Each pixel of the destination image is the pixel with the user-specified rank in the filter window.</p> <p>The source and destination images must be single-channel images.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>dst</i></td><td>Pointer to destination image.</td></tr><tr><td><i>src</i></td><td>Pointer to source image.</td></tr><tr><td><i>rank</i></td><td>The rank of the destination pixel. The pixel with minimum value is designated rank 0.</td></tr></table>	<i>dst</i>	Pointer to destination image.	<i>src</i>	Pointer to source image.	<i>rank</i>	The rank of the destination pixel. The pixel with minimum value is designated rank 0.
<i>dst</i>	Pointer to destination image.						
<i>src</i>	Pointer to source image.						
<i>rank</i>	The rank of the destination pixel. The pixel with minimum value is designated rank 0.						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageMaxFilter3x3(3MLIB)</code> , <code>mllib_ImageMaxFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMaxFilter5x5(3MLIB)</code> , <code>mllib_ImageMaxFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMaxFilter7x7(3MLIB)</code> , <code>mllib_ImageMaxFilter7x7_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3_US(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5_US(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7_US(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN_US(3MLIB)</code> , <code>mllib_ImageMinFilter3x3(3MLIB)</code> , <code>mllib_ImageMinFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMinFilter5x5(3MLIB)</code> , <code>mllib_ImageMinFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMinFilter7x7(3MLIB)</code> , <code>mllib_ImageMinFilter7x7_Fp(3MLIB)</code> , <code>mllib_ImageRankFilter3x3(3MLIB)</code> ,						

mllib_ImageRankFilter7x7_Fp(3MLIB)

```
mllib_ImageRankFilter3x3_Fp(3MLIB),  
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),  
mllib_ImageRankFilter5x5_Fp(3MLIB),  
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),  
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),  
mllib_ImageRankFilterMxN_Fp(3MLIB),  
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

mllib_ImageRankFilter7x7_US(3MLIB)

NAME	mllib_ImageRankFilter7x7_US – 7x7 rank filter, unsigned								
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageRankFilter7x7_US(mllib_image *dst, const mllib_image *src, mllib_s32 rank, mllib_s32 bits);</pre>								
DESCRIPTION	<p>The <code>mllib_ImageRankFilter7x7_US()</code> function performs 7x7 rank filtering on an <code>MLIB_SHORT</code> type of image that contains unsigned data. Each pixel of the destination image is the pixel with the user-specified rank in the filter window.</p> <p>The source and destination images must be single-channel images.</p>								
PARAMETERS	<p>The function takes the following arguments:</p> <table border="0"> <tr> <td style="padding-right: 20px;"><i>dst</i></td> <td>Pointer to destination image.</td> </tr> <tr> <td><i>src</i></td> <td>Pointer to source image.</td> </tr> <tr> <td><i>rank</i></td> <td>The rank of the destination pixel. The pixel with minimum value is designated rank 0.</td> </tr> <tr> <td><i>bits</i></td> <td>The number of unsigned bits for pixel dynamic range. $9 \leq \text{bits} \leq 15$.</td> </tr> </table>	<i>dst</i>	Pointer to destination image.	<i>src</i>	Pointer to source image.	<i>rank</i>	The rank of the destination pixel. The pixel with minimum value is designated rank 0.	<i>bits</i>	The number of unsigned bits for pixel dynamic range. $9 \leq \text{bits} \leq 15$.
<i>dst</i>	Pointer to destination image.								
<i>src</i>	Pointer to source image.								
<i>rank</i>	The rank of the destination pixel. The pixel with minimum value is designated rank 0.								
<i>bits</i>	The number of unsigned bits for pixel dynamic range. $9 \leq \text{bits} \leq 15$.								
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .								
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:								
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe		
ATTRIBUTE TYPE	ATTRIBUTE VALUE								
Interface Stability	Evolving								
MT-Level	MT-Safe								
SEE ALSO	<p><code>mllib_ImageMaxFilter3x3(3MLIB)</code>, <code>mllib_ImageMaxFilter3x3_Fp(3MLIB)</code>, <code>mllib_ImageMaxFilter5x5(3MLIB)</code>, <code>mllib_ImageMaxFilter5x5_Fp(3MLIB)</code>, <code>mllib_ImageMaxFilter7x7(3MLIB)</code>, <code>mllib_ImageMaxFilter7x7_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3_US(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5_US(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7_US(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilterMxN_US(3MLIB)</code>, <code>mllib_ImageMinFilter3x3(3MLIB)</code>,</p>								

mllib_ImageRankFilter7x7_US(3MLIB)

```
mllib_ImageMinFilter3x3_Fp(3MLIB), mllib_ImageMinFilter5x5(3MLIB),  
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7(3MLIB),  
mllib_ImageMinFilter7x7_Fp(3MLIB), mllib_ImageRankFilter3x3(3MLIB),  
mllib_ImageRankFilter3x3_Fp(3MLIB),  
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),  
mllib_ImageRankFilter5x5_Fp(3MLIB),  
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),  
mllib_ImageRankFilter7x7_Fp(3MLIB), mllib_ImageRankFilterMxN(3MLIB),  
mllib_ImageRankFilterMxN_Fp(3MLIB),  
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

mllib_ImageRankFilterMxN(3MLIB)

NAME	mllib_ImageRankFilterMxN – MxN rank filter										
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageRankFilterMxN(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, mllib_s32 <i>m</i>, mllib_s32 <i>n</i>, mllib_s32 <i>rank</i>);</pre>										
DESCRIPTION	<p>The <code>mllib_ImageRankFilterMxN()</code> function performs MxN rank filtering on an image. Each pixel of the destination image is the pixel with the user-specified rank in the filter window.</p> <p>The source and destination images must be single-channel images.</p>										
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>dst</i></td><td>Pointer to destination image.</td></tr><tr><td><i>src</i></td><td>Pointer to source image.</td></tr><tr><td><i>m</i></td><td>Width of the filter window. <i>m</i> must be odd number greater than 1.</td></tr><tr><td><i>n</i></td><td>Height of the filter window. <i>n</i> must be odd number greater than 1.</td></tr><tr><td><i>rank</i></td><td>The rank of the destination pixel. The pixel with minimum value is designated rank 0.</td></tr></table>	<i>dst</i>	Pointer to destination image.	<i>src</i>	Pointer to source image.	<i>m</i>	Width of the filter window. <i>m</i> must be odd number greater than 1.	<i>n</i>	Height of the filter window. <i>n</i> must be odd number greater than 1.	<i>rank</i>	The rank of the destination pixel. The pixel with minimum value is designated rank 0.
<i>dst</i>	Pointer to destination image.										
<i>src</i>	Pointer to source image.										
<i>m</i>	Width of the filter window. <i>m</i> must be odd number greater than 1.										
<i>n</i>	Height of the filter window. <i>n</i> must be odd number greater than 1.										
<i>rank</i>	The rank of the destination pixel. The pixel with minimum value is designated rank 0.										
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .										
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:										
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe				
ATTRIBUTE TYPE	ATTRIBUTE VALUE										
Interface Stability	Evolving										
MT-Level	MT-Safe										
SEE ALSO	<code>mllib_ImageMaxFilter3x3(3MLIB)</code> , <code>mllib_ImageMaxFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMaxFilter5x5(3MLIB)</code> , <code>mllib_ImageMaxFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMaxFilter7x7(3MLIB)</code> , <code>mllib_ImageMaxFilter7x7_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3_US(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5_US(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7_US(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN_US(3MLIB)</code> , <code>mllib_ImageMinFilter3x3(3MLIB)</code> ,										

mllib_ImageRankFilterMxN(3MLIB)

```
mllib_ImageMinFilter3x3_Fp(3MLIB), mllib_ImageMinFilter5x5(3MLIB),  
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7(3MLIB),  
mllib_ImageMinFilter7x7_Fp(3MLIB), mllib_ImageRankFilter3x3(3MLIB),  
mllib_ImageRankFilter3x3_Fp(3MLIB),  
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),  
mllib_ImageRankFilter5x5_Fp(3MLIB),  
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),  
mllib_ImageRankFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter7x7_US(3MLIB),  
mllib_ImageRankFilterMxN_Fp(3MLIB),  
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

mllib_ImageRankFilterMxN_Fp(3MLIB)

NAME	mllib_ImageRankFilterMxN_Fp – MxN rank filter										
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageRankFilterMxN_Fp(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, mllib_s32 <i>m</i>, mllib_s32 <i>n</i>, mllib_s32 <i>rank</i>);</pre>										
DESCRIPTION	<p>The <code>mllib_ImageRankFilterMxN_Fp()</code> function performs floating-point MxN rank filtering on an image. Each pixel of the destination image is the pixel with the user-specified rank in the filter window.</p> <p>The source and destination images must be single-channel images.</p>										
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>dst</i></td><td>Pointer to destination image.</td></tr><tr><td><i>src</i></td><td>Pointer to source image.</td></tr><tr><td><i>m</i></td><td>Width of the filter window. <i>m</i> must be odd number greater than 1.</td></tr><tr><td><i>n</i></td><td>Height of the filter window. <i>n</i> must be odd number greater than 1.</td></tr><tr><td><i>rank</i></td><td>The rank of the destination pixel. The pixel with minimum value is designated rank 0.</td></tr></table>	<i>dst</i>	Pointer to destination image.	<i>src</i>	Pointer to source image.	<i>m</i>	Width of the filter window. <i>m</i> must be odd number greater than 1.	<i>n</i>	Height of the filter window. <i>n</i> must be odd number greater than 1.	<i>rank</i>	The rank of the destination pixel. The pixel with minimum value is designated rank 0.
<i>dst</i>	Pointer to destination image.										
<i>src</i>	Pointer to source image.										
<i>m</i>	Width of the filter window. <i>m</i> must be odd number greater than 1.										
<i>n</i>	Height of the filter window. <i>n</i> must be odd number greater than 1.										
<i>rank</i>	The rank of the destination pixel. The pixel with minimum value is designated rank 0.										
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .										
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:										
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe				
ATTRIBUTE TYPE	ATTRIBUTE VALUE										
Interface Stability	Evolving										
MT-Level	MT-Safe										
SEE ALSO	<code>mllib_ImageMaxFilter3x3(3MLIB)</code> , <code>mllib_ImageMaxFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMaxFilter5x5(3MLIB)</code> , <code>mllib_ImageMaxFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMaxFilter7x7(3MLIB)</code> , <code>mllib_ImageMaxFilter7x7_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter3x3_US(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter5x5_US(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilter7x7_US(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN_Fp(3MLIB)</code> , <code>mllib_ImageMedianFilterMxN_US(3MLIB)</code> , <code>mllib_ImageMinFilter3x3(3MLIB)</code> ,										

`mllib_ImageRankFilterMxN_Fp(3MLIB)`

```
mllib_ImageMinFilter3x3_Fp(3MLIB), mllib_ImageMinFilter5x5(3MLIB),  
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7(3MLIB),  
mllib_ImageMinFilter7x7_Fp(3MLIB), mllib_ImageRankFilter3x3(3MLIB),  
mllib_ImageRankFilter3x3_Fp(3MLIB),  
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),  
mllib_ImageRankFilter5x5_Fp(3MLIB),  
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),  
mllib_ImageRankFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),  
mllib_ImageRankFilterMxN_US(3MLIB), attributes(5)
```

mllib_ImageRankFilterMxN_US(3MLIB)

NAME	mllib_ImageRankFilterMxN_US – MxN rank filter, unsigned												
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageRankFilterMxN_US(mllib_image *dst, const mllib_image *src, mllib_s32 m, mllib_s32 n, mllib_s32 rank, mllib_s32 bits);</pre>												
DESCRIPTION	<p>The <code>mllib_ImageRankFilterMxN_US()</code> function performs MxN rank filtering on an <code>MLIB_SHORT</code> type of image that contains unsigned data. Each pixel of the destination image is the pixel with the user-specified rank in the filter window.</p> <p>The source and destination images must be single-channel images.</p>												
PARAMETERS	<p>The function takes the following arguments:</p> <table border="0"> <tr> <td style="padding-right: 20px;"><i>dst</i></td> <td>Pointer to destination image.</td> </tr> <tr> <td><i>src</i></td> <td>Pointer to source image.</td> </tr> <tr> <td><i>m</i></td> <td>Width of the filter window. <i>m</i> must be odd number greater than 1.</td> </tr> <tr> <td><i>n</i></td> <td>Height of the filter window. <i>n</i> must be odd number greater than 1.</td> </tr> <tr> <td><i>rank</i></td> <td>The rank of the destination pixel. The pixel with minimum value is designated rank 0.</td> </tr> <tr> <td><i>bits</i></td> <td>The number of unsigned bits for pixel dynamic range. $9 \leq bits \leq 15$.</td> </tr> </table>	<i>dst</i>	Pointer to destination image.	<i>src</i>	Pointer to source image.	<i>m</i>	Width of the filter window. <i>m</i> must be odd number greater than 1.	<i>n</i>	Height of the filter window. <i>n</i> must be odd number greater than 1.	<i>rank</i>	The rank of the destination pixel. The pixel with minimum value is designated rank 0.	<i>bits</i>	The number of unsigned bits for pixel dynamic range. $9 \leq bits \leq 15$.
<i>dst</i>	Pointer to destination image.												
<i>src</i>	Pointer to source image.												
<i>m</i>	Width of the filter window. <i>m</i> must be odd number greater than 1.												
<i>n</i>	Height of the filter window. <i>n</i> must be odd number greater than 1.												
<i>rank</i>	The rank of the destination pixel. The pixel with minimum value is designated rank 0.												
<i>bits</i>	The number of unsigned bits for pixel dynamic range. $9 \leq bits \leq 15$.												
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .												
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:												
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe						
ATTRIBUTE TYPE	ATTRIBUTE VALUE												
Interface Stability	Evolving												
MT-Level	MT-Safe												
SEE ALSO	<p><code>mllib_ImageMaxFilter3x3(3MLIB)</code>, <code>mllib_ImageMaxFilter3x3_Fp(3MLIB)</code>, <code>mllib_ImageMaxFilter5x5(3MLIB)</code>, <code>mllib_ImageMaxFilter5x5_Fp(3MLIB)</code>, <code>mllib_ImageMaxFilter7x7(3MLIB)</code>, <code>mllib_ImageMaxFilter7x7_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter3x3_US(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5_Fp(3MLIB)</code>, <code>mllib_ImageMedianFilter5x5_US(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7(3MLIB)</code>, <code>mllib_ImageMedianFilter7x7_Fp(3MLIB)</code>,</p>												

mllib_ImageRankFilterMxN_US(3MLIB)

```
mllib_ImageMedianFilter7x7_US(3MLIB),  
mllib_ImageMedianFilterMxN(3MLIB),  
mllib_ImageMedianFilterMxN_Fp(3MLIB),  
mllib_ImageMedianFilterMxN_US(3MLIB), mllib_ImageMinFilter3x3(3MLIB),  
mllib_ImageMinFilter3x3_Fp(3MLIB), mllib_ImageMinFilter5x5(3MLIB),  
mllib_ImageMinFilter5x5_Fp(3MLIB), mllib_ImageMinFilter7x7(3MLIB),  
mllib_ImageMinFilter7x7_Fp(3MLIB), mllib_ImageRankFilter3x3(3MLIB),  
mllib_ImageRankFilter3x3_Fp(3MLIB),  
mllib_ImageRankFilter3x3_US(3MLIB), mllib_ImageRankFilter5x5(3MLIB),  
mllib_ImageRankFilter5x5_Fp(3MLIB),  
mllib_ImageRankFilter5x5_US(3MLIB), mllib_ImageRankFilter7x7(3MLIB),  
mllib_ImageRankFilter7x7_Fp(3MLIB),  
mllib_ImageRankFilter7x7_US(3MLIB), mllib_ImageRankFilterMxN(3MLIB),  
mllib_ImageRankFilterMxN_Fp(3MLIB), attributes(5)
```

mllib_ImageReformat(3MLIB)

NAME	mllib_ImageReformat – image data buffer reformat																								
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageReformat(void **dstData, const void **srcData, mllib_s32 numBands, mllib_s32 xSize, mllib_s32 ySize, mllib_type dstDataType, const mllib_s32 *dstBandoffsets, mllib_s32 dstScanlinestride, mllib_s32 dstPixelstride, mllib_type srcDataType, const mllib_s32 *srcBandoffsets, mllib_s32 srcScanlinestride, mllib_s32 srcPixelstride);</pre>																								
DESCRIPTION	<p>The <code>mllib_ImageReformat()</code> function copies and casts, if needed, an image from one buffer to another. The formats and data types of the two buffers may be different.</p> <pre>dstPixel[x][y][i] = (dstDataType) srcPixel[x][y][i]</pre> <p>where the values of a pixel at position (x, y) and in channel i are:</p> <pre>srcPixel[x][y][i] = srcData[i][srcBandoffsets[i] + srcScanlinestride*y + srcPixelstride*x] dstPixel[x][y][i] = dstData[i][dstBandoffsets[i] + dstScanlinestride*y + dstPixelstride*x]</pre> <p>It is the user's responsibility to make sure that the data buffers supplied are suitable for this operation. The <code>srcData</code> and <code>dstData</code> can have 1, 2, 3, or 4 channels, and they must have the same number of channels. The <code>srcDataType</code> and <code>dstDataType</code> can be <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, <code>MLIB_INT</code>, <code>MLIB_FLOAT</code>, or <code>MLIB_DOUBLE</code>.</p> <p>The conversions between different data types are implemented as described in the following table:</p> <table><thead><tr><th>Source Type</th><th>Dest. Type</th><th>Action</th></tr></thead><tbody><tr><td>MLIB_SHORT</td><td>MLIB_BYTE</td><td>(mllib_u8)clamp(x, 0, 255)</td></tr><tr><td>MLIB_USHORT</td><td>MLIB_BYTE</td><td>(mllib_u8)clamp(x, 0, 255)</td></tr><tr><td>MLIB_INT</td><td>MLIB_BYTE</td><td>(mllib_u8)clamp(x, 0, 255)</td></tr><tr><td>MLIB_FLOAT</td><td>MLIB_BYTE</td><td>(mllib_u8)clamp(x, 0, 255)</td></tr><tr><td>MLIB_DOUBLE</td><td>MLIB_BYTE</td><td>(mllib_u8)clamp(x, 0, 255)</td></tr><tr><td>MLIB_BYTE</td><td>MLIB_SHORT</td><td>(mllib_s16)x</td></tr><tr><td>MLIB_USHORT</td><td>MLIB_SHORT</td><td>(mllib_s16)clamp(x, -32768, 32767)</td></tr></tbody></table>	Source Type	Dest. Type	Action	MLIB_SHORT	MLIB_BYTE	(mllib_u8)clamp(x, 0, 255)	MLIB_USHORT	MLIB_BYTE	(mllib_u8)clamp(x, 0, 255)	MLIB_INT	MLIB_BYTE	(mllib_u8)clamp(x, 0, 255)	MLIB_FLOAT	MLIB_BYTE	(mllib_u8)clamp(x, 0, 255)	MLIB_DOUBLE	MLIB_BYTE	(mllib_u8)clamp(x, 0, 255)	MLIB_BYTE	MLIB_SHORT	(mllib_s16)x	MLIB_USHORT	MLIB_SHORT	(mllib_s16)clamp(x, -32768, 32767)
Source Type	Dest. Type	Action																							
MLIB_SHORT	MLIB_BYTE	(mllib_u8)clamp(x, 0, 255)																							
MLIB_USHORT	MLIB_BYTE	(mllib_u8)clamp(x, 0, 255)																							
MLIB_INT	MLIB_BYTE	(mllib_u8)clamp(x, 0, 255)																							
MLIB_FLOAT	MLIB_BYTE	(mllib_u8)clamp(x, 0, 255)																							
MLIB_DOUBLE	MLIB_BYTE	(mllib_u8)clamp(x, 0, 255)																							
MLIB_BYTE	MLIB_SHORT	(mllib_s16)x																							
MLIB_USHORT	MLIB_SHORT	(mllib_s16)clamp(x, -32768, 32767)																							

mllib_ImageReformat(3MLIB)

Source Type	Dest. Type	Action
MLIB_INT	MLIB_SHORT	(mllib_s16)clamp(x, -32768, 32767)
MLIB_FLOAT	MLIB_SHORT	(mllib_s16)clamp(x, -32768, 32767)
MLIB_DOUBLE	MLIB_SHORT	(mllib_s16)clamp(x, -32768, 32767)
MLIB_BYTE	MLIB_USHORT	(mllib_u16)x
MLIB_SHORT	MLIB_USHORT	(mllib_u16)clamp(x, 0, 65535)
MLIB_INT	MLIB_USHORT	(mllib_u16)clamp(x, 0, 65535)
MLIB_FLOAT	MLIB_USHORT	(mllib_u16)clamp(x, 0, 65535)
MLIB_DOUBLE	MLIB_USHORT	(mllib_u16)clamp(x, 0, 65535)
MLIB_BYTE	MLIB_INT	(mllib_s32)x
MLIB_SHORT	MLIB_INT	(mllib_s32)x
MLIB_USHORT	MLIB_INT	(mllib_s32)x
MLIB_FLOAT	MLIB_INT	(mllib_s32)clamp(x, -2147483647-1, 2147483647)
MLIB_DOUBLE	MLIB_INT	(mllib_s32)clamp(x, -2147483647-1, 2147483647)
MLIB_BYTE	MLIB_FLOAT	(mllib_f32)x
MLIB_SHORT	MLIB_FLOAT	(mllib_f32)x
MLIB_USHORT	MLIB_FLOAT	(mllib_f32)x
MLIB_INT	MLIB_FLOAT	(mllib_f32)x
MLIB_DOUBLE	MLIB_FLOAT	(mllib_f32)x
MLIB_BYTE	MLIB_DOUBLE	(mllib_d64)x
MLIB_SHORT	MLIB_DOUBLE	(mllib_d64)x
MLIB_USHORT	MLIB_DOUBLE	(mllib_d64)x
MLIB_INT	MLIB_DOUBLE	(mllib_d64)x
MLIB_FLOAT	MLIB_DOUBLE	(mllib_d64)x

The actions are defined in C-style pseudo-code. All type casts follow the rules of standard C. `clamp()` can be defined as a macro: `#define clamp(x, low, high) (((x) < (low)) ? (low) : (((x) > (high)) ? (high) : (x)))`

mllib_ImageReformat(3MLIB)

PARAMETERS The function takes the following arguments:

<i>dstData</i>	The pointer to the destination image data buffer.
<i>srcData</i>	The pointer to the source image data buffer.
<i>numBands</i>	The number of channels of the image data buffers.
<i>xSize</i>	The width of the image.
<i>ySize</i>	The height of the image.
<i>dstDataType</i>	The data type of the <i>dstData</i> buffer.
<i>dstBandoffsets</i>	The initial pixel's offsets in the <i>dstData</i> buffer in terms of destination data buffer elements.
<i>dstScanlinestride</i>	The scanline stride of the <i>dstData</i> buffer in terms of destination data buffer elements.
<i>dstPixelstride</i>	The pixel stride of the <i>dstData</i> buffer in terms of destination data buffer elements.
<i>srcDataType</i>	The data type of the <i>srcData</i> buffer.
<i>srcBandoffsets</i>	The initial pixel's offsets in the <i>srcData</i> buffer in terms of source data buffer elements.
<i>srcScanlinestride</i>	The scanline stride of the <i>srcData</i> buffer in terms of source data buffer elements.
<i>srcPixelstride</i>	The pixel stride of the <i>srcData</i> buffer in terms of source data buffer elements.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_ImageDataTypeConvert\(3MLIB\)](#), [attributes\(5\)](#)

NAME | mllib_ImageReplaceColor – replace a color in an image

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageReplaceColor(mllib_image *dst, const
    mllib_image *src, const mllib_s32 *color1, const mllib_s32 *color2);
```

DESCRIPTION | The mllib_ImageReplaceColor() function copies the source image to the destination image and replaces the pixels having a value of *color1* with *color2*.
It uses the following equation:

$$\begin{aligned} \text{dst}[x][y] &= \text{color2} && \text{if } \text{src}[x][y] == \text{color1} \\ \text{dst}[x][y] &= \text{src}[x][y] && \text{if } \text{src}[x][y] \neq \text{color1} \end{aligned}$$

PARAMETERS | The function takes the following arguments:

dst | Pointer to destination image.

src | Pointer to source image.

color1 | Array of color components to be replaced.

color2 | Array of color components to replace *color1*.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_ImageReplaceColor_Inp(3MLIB),
mllib_ImageReplaceColor_Fp(3MLIB),
mllib_ImageReplaceColor_Fp_Inp(3MLIB), mllib_ImageThresh5(3MLIB),
mllib_ImageThresh5_Inp(3MLIB), mllib_ImageThresh5_Fp(3MLIB),
mllib_ImageThresh5_Fp_Inp(3MLIB), attributes(5)

mllib_ImageReplaceColor_Fp(3MLIB)

NAME	mllib_ImageReplaceColor_Fp – replace a color in an image						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageReplaceColor_Fp(mllib_image *dst, const mllib_image *src, const mllib_d64 *color1, const mllib_d64 *color2);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageReplaceColor_Fp()</code> function copies the source image to the destination image and replaces the pixels having a value of <code>color1</code> with <code>color2</code>.</p> <p>It uses the following equation:</p> $\begin{aligned} \text{dst}[x][y] &= \text{color2} && \text{if } \text{src}[x][y] == \text{color1} \\ \text{dst}[x][y] &= \text{src}[x][y] && \text{if } \text{src}[x][y] \neq \text{color1} \end{aligned}$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>color1</i> Array of color components to be replaced.</p> <p><i>color2</i> Array of color components to replace <code>color1</code>.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageReplaceColor(3MLIB)</code> , <code>mllib_ImageReplaceColor_Inp(3MLIB)</code> , <code>mllib_ImageReplaceColor_Fp_Inp(3MLIB)</code> , <code>mllib_ImageThresh5(3MLIB)</code> , <code>mllib_ImageThresh5_Inp(3MLIB)</code> , <code>mllib_ImageThresh5_Fp(3MLIB)</code> , <code>mllib_ImageThresh5_Fp_Inp(3MLIB)</code> , <code>attributes(5)</code>						

mlib_ImageReplaceColor_Fp_Inp(3MLIB)

NAME | mlib_ImageReplaceColor_Fp_Inp – replace a color in an image, in place

SYNOPSIS |

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_ImageReplaceColor_Fp_Inp(mlib_image *srcdst,
      const mlib_d64 *color1, const mlib_d64 *color2);
```

DESCRIPTION | The `mlib_ImageReplaceColor_Fp_Inp()` function scans the image for all pixels with color value equal to *color1* and replaces these pixels with *color2*.

It uses the following equation:

```
srcdst[x][y] = color2 if srcdst[x][y] == color1
```

PARAMETERS | The function takes the following arguments:

srcdst | Pointer to the source and destination image.

color1 | Array of color components to be replaced.

color2 | Array of color components to replace *color1*.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mlib_ImageReplaceColor(3MLIB)`, `mlib_ImageReplaceColor_Inp(3MLIB)`, `mlib_ImageReplaceColor_Fp(3MLIB)`, `mlib_ImageThresh5(3MLIB)`, `mlib_ImageThresh5_Inp(3MLIB)`, `mlib_ImageThresh5_Fp(3MLIB)`, `mlib_ImageThresh5_Fp_Inp(3MLIB)`, `attributes(5)`

mllib_ImageReplaceColor_Inp(3MLIB)

NAME	mllib_ImageReplaceColor_Inp – replace a color in an image, in place						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageReplaceColor_Inp(mllib_image *srcdst, const mllib_s32 *color1, const mllib_s32 *color2);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageReplaceColor_Inp()</code> function scans the image for all pixels with color value equal to <code>color1</code> and replaces these pixels with <code>color2</code>.</p> <p>It uses the following equation:</p> $\text{srcdst}[x][y] = \text{color2} \quad \text{if } \text{srcdst}[x][y] == \text{color1}$						
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>srcdst</i></td><td>Pointer to the source and destination image.</td></tr><tr><td><i>color1</i></td><td>Array of color components to be replaced.</td></tr><tr><td><i>color2</i></td><td>Array of color components to replace <code>color1</code>.</td></tr></table>	<i>srcdst</i>	Pointer to the source and destination image.	<i>color1</i>	Array of color components to be replaced.	<i>color2</i>	Array of color components to replace <code>color1</code> .
<i>srcdst</i>	Pointer to the source and destination image.						
<i>color1</i>	Array of color components to be replaced.						
<i>color2</i>	Array of color components to replace <code>color1</code> .						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageReplaceColor(3MLIB)</code> , <code>mllib_ImageReplaceColor_Fp(3MLIB)</code> , <code>mllib_ImageReplaceColor_Fp_Inp(3MLIB)</code> , <code>mllib_ImageThresh5(3MLIB)</code> , <code>mllib_ImageThresh5_Inp(3MLIB)</code> , <code>mllib_ImageThresh5_Fp(3MLIB)</code> , <code>mllib_ImageThresh5_Fp_Inp(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageRotate180 – rotate an image by 180 degrees

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageRotate180(mllib_image *dst, const mllib_image
    *src);
```

DESCRIPTION Rotate an image 180 degrees.

The width and height of the destination image can be different from the width and height of the source image. The center of the source image is mapped to the center of the destination image.

The data type of the images can be MLIB_BIT, MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, or MLIB_INT.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageFlipAntiDiag(3MLIB), mllib_ImageFlipAntiDiag_Fp(3MLIB), mllib_ImageFlipMainDiag(3MLIB), mllib_ImageFlipMainDiag_Fp(3MLIB), mllib_ImageFlipX(3MLIB), mllib_ImageFlipX_Fp(3MLIB), mllib_ImageFlipY(3MLIB), mllib_ImageFlipY_Fp(3MLIB), mllib_ImageRotate90(3MLIB), mllib_ImageRotate90_Fp(3MLIB), mllib_ImageRotate180_Fp(3MLIB), mllib_ImageRotate270(3MLIB), mllib_ImageRotate270_Fp(3MLIB), attributes(5)

mllib_ImageRotate180_Fp(3MLIB)

NAME | mllib_ImageRotate180_Fp – rotate an image by 180 degrees

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
mllib_status mllib_ImageRotate180_Fp(mllib_image *dst, const  
mllib_image *src);
```

DESCRIPTION | Rotate a floating-point image 180 degrees.

The width and height of the destination image can be different from the width and height of the source image. The center of the source image is mapped to the center of the destination image.

PARAMETERS | The function takes the following arguments:

dst | Pointer to destination image.

src | Pointer to source image.

RETURN VALUES | The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_ImageFlipAntiDiag(3MLIB), mllib_ImageFlipAntiDiag_Fp(3MLIB), mllib_ImageFlipMainDiag(3MLIB), mllib_ImageFlipMainDiag_Fp(3MLIB), mllib_ImageFlipX(3MLIB), mllib_ImageFlipX_Fp(3MLIB), mllib_ImageFlipY(3MLIB), mllib_ImageFlipY_Fp(3MLIB), mllib_ImageRotate90(3MLIB), mllib_ImageRotate90_Fp(3MLIB), mllib_ImageRotate180(3MLIB), mllib_ImageRotate270(3MLIB), mllib_ImageRotate270_Fp(3MLIB), attributes(5)

NAME mllib_ImageRotate270 – rotate an image by 270 degrees

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageRotate270(mllib_image *dst, const mllib_image
    *src);
```

DESCRIPTION Rotate an image 270 degrees clockwise.

The width and height of the destination image can be different from the width and height of the source image. The center of the source image is mapped to the center of the destination image.

The data type of the images can be MLIB_BIT, MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, or MLIB_INT.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageFlipAntiDiag(3MLIB), mllib_ImageFlipAntiDiag_Fp(3MLIB), mllib_ImageFlipMainDiag(3MLIB), mllib_ImageFlipMainDiag_Fp(3MLIB), mllib_ImageFlipX(3MLIB), mllib_ImageFlipX_Fp(3MLIB), mllib_ImageFlipY(3MLIB), mllib_ImageFlipY_Fp(3MLIB), mllib_ImageRotate90(3MLIB), mllib_ImageRotate90_Fp(3MLIB), mllib_ImageRotate180(3MLIB), mllib_ImageRotate180_Fp(3MLIB), mllib_ImageRotate270_Fp(3MLIB), attributes(5)

mllib_ImageRotate270_Fp(3MLIB)

NAME | mllib_ImageRotate270_Fp – rotate an image by 270 degrees

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
mllib_status mllib_ImageRotate270_Fp(mllib_image *dst, const  
mllib_image *src);
```

DESCRIPTION | Rotate a floating-point image 270 degrees clockwise.

The width and height of the destination image can be different from the width and height of the source image. The center of the source image is mapped to the center of the destination image.

PARAMETERS | The function takes the following arguments:

dst | Pointer to destination image.

src | Pointer to source image.

RETURN VALUES | The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_ImageFlipAntiDiag(3MLIB), mllib_ImageFlipAntiDiag_Fp(3MLIB), mllib_ImageFlipMainDiag(3MLIB), mllib_ImageFlipMainDiag_Fp(3MLIB), mllib_ImageFlipX(3MLIB), mllib_ImageFlipX_Fp(3MLIB), mllib_ImageFlipY(3MLIB), mllib_ImageFlipY_Fp(3MLIB), mllib_ImageRotate90(3MLIB), mllib_ImageRotate90_Fp(3MLIB), mllib_ImageRotate180(3MLIB), mllib_ImageRotate180_Fp(3MLIB), mllib_ImageRotate270(3MLIB), attributes(5)

NAME	mllib_ImageRotate – rotate image
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageRotate(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, mllib_d64 <i>angle</i>, mllib_d64 <i>xcenter</i>, mllib_d64 <i>ycenter</i>, mllib_filter <i>filter</i>, mllib_edge <i>edge</i>);</pre>
DESCRIPTION	<p>The <code>mllib_ImageRotate()</code> function rotates a source image around a user-defined rotation center in the user-defined radians.</p> <p>The width and height of the destination image can be different from the width and height of the source image. The (<code>xcenter</code>, <code>ycenter</code>) point of the source image is mapped to the center of the destination image. You should ensure that the destination buffer is large enough to hold the resulting bounding box to avoid clipping part of the image.</p> <p>The data type of the images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p> <p>The center of the upper-left corner pixel of an image is located at (0.5, 0.5).</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>angle</i> Angle of rotation. The angle is measured in radians clockwise.</p> <p><i>xcenter</i> X coordinate of rotation center in the source image.</p> <p><i>ycenter</i> Y coordinate of rotation center in the source image.</p> <p><i>filter</i> Type of resampling filter. It can be one of the following:</p> <pre>MLIB_NEAREST MLIB_BILINEAR MLIB_BICUBIC MLIB_BICUBIC2</pre> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_OP_NEAREST MLIB_EDGE_SRC_EXTEND MLIB_EDGE_SRC_PADDED</pre>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .

`mlib_ImageRotate(3MLIB)`

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mlib_ImageRotate_Fp(3MLIB)`, `mlib_ImageRotateIndex(3MLIB)`,
`attributes(5)`

NAME | mllib_ImageRotate90 – rotate an image by 90 degrees

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageRotate90(mllib_image *dst, const mllib_image
    *src);
```

DESCRIPTION | Rotate an image 90 degrees clockwise.

The width and height of the destination image can be different from the width and height of the source image. The center of the source image is mapped to the center of the destination image.

The data type of the images can be MLIB_BIT, MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, or MLIB_INT.

PARAMETERS | The function takes the following arguments:

dst | Pointer to destination image.

src | Pointer to source image.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_ImageFlipAntiDiag(3MLIB), mllib_ImageFlipAntiDiag_Fp(3MLIB), mllib_ImageFlipMainDiag(3MLIB), mllib_ImageFlipMainDiag_Fp(3MLIB), mllib_ImageFlipX(3MLIB), mllib_ImageFlipX_Fp(3MLIB), mllib_ImageFlipY(3MLIB), mllib_ImageFlipY_Fp(3MLIB), mllib_ImageRotate90_Fp(3MLIB), mllib_ImageRotate180(3MLIB), mllib_ImageRotate180_Fp(3MLIB), mllib_ImageRotate270(3MLIB), mllib_ImageRotate270_Fp(3MLIB), attributes(5)

mllib_ImageRotate90_Fp(3MLIB)

NAME | mllib_ImageRotate90_Fp – rotate an image by 90 degrees

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageRotate90_Fp(mllib_image *dst, const
mllib_image *src);
```

DESCRIPTION | Rotate a floating-point image 90 degrees clockwise.

The width and height of the destination image can be different from the width and height of the source image. The center of the source image is mapped to the center of the destination image.

PARAMETERS | The function takes the following arguments:

dst | Pointer to destination image.

src | Pointer to source image.

RETURN VALUES | The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_ImageFlipAntiDiag(3MLIB), mllib_ImageFlipAntiDiag_Fp(3MLIB), mllib_ImageFlipMainDiag(3MLIB), mllib_ImageFlipMainDiag_Fp(3MLIB), mllib_ImageFlipX(3MLIB), mllib_ImageFlipX_Fp(3MLIB), mllib_ImageFlipY(3MLIB), mllib_ImageFlipY_Fp(3MLIB), mllib_ImageRotate90(3MLIB), mllib_ImageRotate180(3MLIB), mllib_ImageRotate180_Fp(3MLIB), mllib_ImageRotate270(3MLIB), mllib_ImageRotate270_Fp(3MLIB), attributes(5)

NAME	mllib_ImageRotate_Fp – rotate image
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageRotate_Fp(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, mllib_d64 <i>angle</i>, mllib_d64 <i>xcenter</i>, mllib_d64 <i>ycenter</i>, mllib_filter <i>filter</i>, mllib_edge <i>edge</i>);</pre>
DESCRIPTION	<p>The <code>mllib_ImageRotate_Fp()</code> function rotates a floating-point source image around a user-defined rotation center in the user-defined radians.</p> <p>The width and height of the destination image can be different from the width and height of the source image. The (<code>xcenter</code>, <code>ycenter</code>) point of the source image is mapped to the center of the destination image. You should ensure that the destination buffer is large enough to hold the resulting bounding box to avoid clipping part of the image.</p> <p>The center of the upper-left corner pixel of an image is located at (0.5, 0.5).</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>angle</i> Angle of rotation. The angle is measured in radians clockwise.</p> <p><i>xcenter</i> X coordinate of rotation center in the source image.</p> <p><i>ycenter</i> Y coordinate of rotation center in the source image.</p> <p><i>filter</i> Type of resampling filter. It can be one of the following:</p> <pre>MLIB_NEAREST MLIB_BILINEAR MLIB_BICUBIC MLIB_BICUBIC2</pre> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_OP_NEAREST MLIB_EDGE_SRC_EXTEND MLIB_EDGE_SRC_PADDED</pre>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:

mllib_ImageRotate_Fp(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_ImageRotate\(3MLIB\)](#), [mllib_ImageRotateIndex\(3MLIB\)](#), [attributes\(5\)](#)

NAME	mllib_ImageRotateIndex – rotate color-indexed image
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageRotateIndex(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, mllib_d64 <i>angle</i>, mllib_d64 <i>xcenter</i>, mllib_d64 <i>ycenter</i>, mllib_filter <i>filter</i>, mllib_edge <i>edge</i>, const void *<i>colormap</i>);</pre>
DESCRIPTION	<p>The <code>mllib_ImageRotateIndex()</code> function rotates the source image about a user-defined rotation center in user-defined radians.</p> <p>The width and height of the destination image can be different from the width and height of the source image. The (<i>xcenter</i>, <i>ycenter</i>) point of the source image is mapped to the center of the destination image. You should ensure that the destination buffer is large enough to hold the resulting bounding box to avoid clipping part of the image.</p> <p>The source and destination images must be single-channel images.</p> <p>The image data type must be <code>MLIB_BYTE</code> or <code>MLIB_SHORT</code>.</p> <p>The center of the upper-left corner pixel of an image is located at (0.5, 0.5).</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>angle</i> Angle of rotation. The angle is measured in radians clockwise.</p> <p><i>xcenter</i> X coordinate of rotation center in the source image.</p> <p><i>ycenter</i> Y coordinate of rotation center in the source image.</p> <p><i>filter</i> Type of resampling filter. It can be one of the following:</p> <pre>MLIB_NEAREST MLIB_BILINEAR MLIB_BICUBIC MLIB_BICUBIC2</pre> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_OP_NEAREST MLIB_EDGE_SRC_EXTEND MLIB_EDGE_SRC_PADDED</pre> <p><i>colormap</i> Internal data structure for inverse color mapping. This data structure is generated by the <code>mllib_ImageColorTrue2IndexInit()</code> function.</p>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .

`mlib_ImageRotateIndex(3MLIB)`

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mlib_ImageRotate(3MLIB)`, `mlib_ImageRotate_Fp(3MLIB)`, `attributes(5)`

NAME mllib_ImageScalarBlend – image blending with scalar

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageScalarBlend(mllib_image dst, const
    mllib_image *src1, const mllib_image *src2, const mllib_s32 *alpha);
```

DESCRIPTION The mllib_ImageScalarBlend() function blends the first and second source images by adding each of their scaled pixels. The first source image is scaled by the scalar *a*, and the second source image is inverse scaled by (1 - *a*).

It uses the following equation:

$$dst[x][y][i] = a[i]*src1[x][y][i] + (1 - a[i])*src2[x][y][i]$$

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src1 Pointer to first source image.

src2 Pointer to second source image.

alpha Scalar blending factor. The *a* value equals (alpha * 2**(-31)). *alpha*[*i*] contains the blending factor for channel *i*.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageScalarBlend_Fp(3MLIB),
 mllib_ImageScalarBlend_Fp_Inp(3MLIB),
 mllib_ImageScalarBlend_Inp(3MLIB), attributes(5)

mllib_ImageScalarBlend_Fp(3MLIB)

NAME	mllib_ImageScalarBlend_Fp – image blending with scalar								
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageScalarBlend_Fp(mllib_image *dst, const mllib_image *src1, const mllib_image *src2, const mllib_d64 *alpha);</pre>								
DESCRIPTION	<p>The <code>mllib_ImageScalarBlend_Fp()</code> function blends the first and second floating-point source images by adding each of their scaled pixels. The first source image is scaled by the scalar <code>a</code>, and the second source image is inverse scaled by $(1 - a)$.</p> <p>It uses the following equation:</p> $dst[x][y][i] = a[i]*src1[x][y][i] + (1 - a[i])*src2[x][y][i]$								
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>dst</i></td><td>Pointer to destination image.</td></tr><tr><td><i>src1</i></td><td>Pointer to first source image.</td></tr><tr><td><i>src2</i></td><td>Pointer to second source image.</td></tr><tr><td><i>alpha</i></td><td>Scalar blending factor. The <code>a</code> value equals <code>alpha</code> which should be in the <code>[0.0, 1.0]</code> range. <code>alpha[i]</code> contains the blending factor for channel <code>i</code>.</td></tr></table>	<i>dst</i>	Pointer to destination image.	<i>src1</i>	Pointer to first source image.	<i>src2</i>	Pointer to second source image.	<i>alpha</i>	Scalar blending factor. The <code>a</code> value equals <code>alpha</code> which should be in the <code>[0.0, 1.0]</code> range. <code>alpha[i]</code> contains the blending factor for channel <code>i</code> .
<i>dst</i>	Pointer to destination image.								
<i>src1</i>	Pointer to first source image.								
<i>src2</i>	Pointer to second source image.								
<i>alpha</i>	Scalar blending factor. The <code>a</code> value equals <code>alpha</code> which should be in the <code>[0.0, 1.0]</code> range. <code>alpha[i]</code> contains the blending factor for channel <code>i</code> .								
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .								
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:								
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe		
ATTRIBUTE TYPE	ATTRIBUTE VALUE								
Interface Stability	Evolving								
MT-Level	MT-Safe								
SEE ALSO	<code>mllib_ImageScalarBlend(3MLIB)</code> , <code>mllib_ImageScalarBlend_Fp_Inp(3MLIB)</code> , <code>mllib_ImageScalarBlend_Inp(3MLIB)</code> , <code>attributes(5)</code>								

NAME mllib_ImageScalarBlend_Fp_Inp – image blending with scalar

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageScalarBlend_Fp_Inp(mllib_image *src1dst, const
mllib_image *src2, const mllib_d64 *alpha);
```

DESCRIPTION The `mllib_ImageScalarBlend_Fp_Inp()` function blends the first and second floating-point source images by adding each of their scaled pixels in place. The first source image is scaled by the scalar `a`, and the second source image is inverse scaled by `(1 - a)`.

It uses the following equation:

$$\text{src1dst}[x][y][i] = a[i] * \text{src1dst}[x][y][i] + (1 - a[i]) * \text{src2}[x][y][i]$$

PARAMETERS The function takes the following arguments:

`src1dst` Pointer to first source and destination image.

`src2` Pointer to second source image.

`alpha` Scalar blending factor. The `a` value equals `alpha` which should be in the `[0.0, 1.0]` range. `alpha[i]` contains the blending factor for channel `i`.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageScalarBlend(3MLIB)`, `mllib_ImageScalarBlend_Fp(3MLIB)`, `mllib_ImageScalarBlend_Inp(3MLIB)`, `attributes(5)`

mllib_ImageScalarBlend_Inp(3MLIB)

NAME	mllib_ImageScalarBlend_Inp – image blending with scalar, in place						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageScalarBlend_Inp(mllib_image *src1dst, const mllib_image *src2, const mllib_s32 *alpha);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageScalarBlend_Inp()</code> function blends the first and second source images by adding each of their scaled pixels in place. The first source image is scaled by the scalar <code>a</code>, and the second source image is inverse scaled by $(1 - a)$.</p> <p>It uses the following equation:</p> $\text{src1dst}[x][y][i] = a[i] * \text{src1dst}[x][y][i] + (1 - a[i]) * \text{src2}[x][y][i]$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>src1dst</i> Pointer to first source and destination image.</p> <p><i>src2</i> Pointer to second source image.</p> <p><i>alpha</i> Scalar blending factor. The <code>a</code> value equals $(\text{alpha} * 2^{**(-31)})$. <code>alpha[i]</code> contains the blending factor for channel <code>i</code>.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageScalarBlend(3MLIB)</code> , <code>mllib_ImageScalarBlend_Fp(3MLIB)</code> , <code>mllib_ImageScalarBlend_Fp_Inp(3MLIB)</code> , <code>attributes(5)</code>						

NAME	mllib_ImageScale2 – linear scaling																																			
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageScale2(mllib_image *dst, const mllib_image *src, const mllib_d64 *alpha, const mllib_d64 *beta);</pre>																																			
DESCRIPTION	<p>The <code>mllib_ImageScale2()</code> function performs a linear scaling on the pixels of the source image by multiplying the data by a scale factor and then adding an offset. Images must have the same size, and number of channels. They can have 1, 2, 3, or 4 channels.</p> <p>The following equation is used:</p> $\text{dst}[x][y][i] = \text{src}[x][y][i] * \text{alpha}[i] + \text{beta}[i]$ <p>If the result of the operation underflows/overflows the minimum/maximum value supported by the destination image, then it will be clamped to the minimum/maximum value respectively.</p> <p>See the following table for available variations of this linear scaling function.</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th>Type [*]</th> <th>BYTE</th> <th>SHORT</th> <th>USHORT</th> <th>INT</th> <th>FLOAT</th> <th>DOUBLE</th> </tr> </thead> <tbody> <tr> <td>MLIB_BYTE</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> </tr> <tr> <td>MLIB_SHORT</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> </tr> <tr> <td>MLIB_USHORT</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> </tr> <tr> <td>MLIB_INT</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> </tr> </tbody> </table> <p>[*] Each row represents a source data type. Each column represents a destination data type.</p>	Type [*]	BYTE	SHORT	USHORT	INT	FLOAT	DOUBLE	MLIB_BYTE	Y	Y	Y	Y	Y	Y	MLIB_SHORT	Y	Y	Y	Y	Y	Y	MLIB_USHORT	Y	Y	Y	Y	Y	Y	MLIB_INT	Y	Y	Y	Y	Y	Y
Type [*]	BYTE	SHORT	USHORT	INT	FLOAT	DOUBLE																														
MLIB_BYTE	Y	Y	Y	Y	Y	Y																														
MLIB_SHORT	Y	Y	Y	Y	Y	Y																														
MLIB_USHORT	Y	Y	Y	Y	Y	Y																														
MLIB_INT	Y	Y	Y	Y	Y	Y																														
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>alpha</i> Scaling factor. <code>alpha[i]</code> contains the scaling factor for channel <i>i</i>.</p> <p><i>beta</i> Offset value. <code>beta[i]</code> contains the offset for channel <i>i</i>.</p>																																			
RETURN VALUES	<p>The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code>.</p>																																			

mllib_ImageScale2(3MLIB)

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageScale(3MLIB)`, `mllib_ImageScale_Fp(3MLIB)`,
`mllib_ImageScale_Fp_Inp(3MLIB)`, `mllib_ImageScale_Inp(3MLIB)`,
`mllib_ImageScale2_Inp(3MLIB)`, `attributes(5)`

NAME mllib_ImageScale2_Inp – linear scaling, in place

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageScale2_Inp(mllib_image *srcdst, const mllib_d64
    *alpha, const mllib_d64 *beta);
```

DESCRIPTION The mllib_ImageScale2_Inp() function performs an in-place linear scaling on the pixels of the source image by multiplying the data by a scale factor and then adding an offset. Images can have 1, 2, 3, or 4 channels.

The following equation is used:

$$\text{srcdst}[x][y][i] = \text{srcdst}[x][y][i] * \text{alpha}[i] + \text{beta}[i]$$

If the result of the operation underflows/overflows the minimum/maximum value supported by the destination image, then it will be clamped to the minimum/maximum value respectively.

The image can be of type MLIB_BYTE, MLIB_SHORT, MLIB_USHORT or MLIB_INT.

PARAMETERS The function takes the following arguments:

srcdst Pointer to source and destination image.

alpha Scaling factor. alpha[i] contains the scaling factor for channel i.

beta Offset value. beta[i] contains the offset for channel i.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageScale(3MLIB), mllib_ImageScale_Fp(3MLIB), mllib_ImageScale_Fp_Inp(3MLIB), mllib_ImageScale_Inp(3MLIB), mllib_ImageScale2(3MLIB), attributes(5)

mllib_ImageScale(3MLIB)

NAME	mllib_ImageScale – linear scaling																																			
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageScale(mllib_image *dst, const mllib_image *src, const mllib_s32 *alpha, const mllib_s32 *beta, mllib_s32 shift);</pre>																																			
DESCRIPTION	<p>The <code>mllib_ImageScale()</code> function performs a linear scaling on the pixels of the source image by multiplying the data by a scale factor, shifting, and then adding an offset.</p> <p>The following equation is used:</p> $\text{dst}[x][y][i] = \text{src}[x][y][i] * \text{alpha}[i] * 2^{*(-\text{shift})} + \text{beta}[i]$ <p>If the result of the operation underflows/overflows the minimum/maximum value supported by the destination image, then it will be clamped to the minimum/maximum value respectively.</p> <p>See the following table for available variations of this linear scaling function.</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th>Type [*]</th> <th>BYTE</th> <th>SHORT</th> <th>USHORT</th> <th>INT</th> <th>FLOAT</th> <th>DOUBLE</th> </tr> </thead> <tbody> <tr> <td>MLIB_BYTE</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> </tr> <tr> <td>MLIB_SHORT</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> </tr> <tr> <td>MLIB_USHORT</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> </tr> <tr> <td>MLIB_INT</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> </tr> </tbody> </table> <p>[*] Each row represents a source data type. Each column represents a destination data type.</p>	Type [*]	BYTE	SHORT	USHORT	INT	FLOAT	DOUBLE	MLIB_BYTE	Y	Y	Y	Y	Y	Y	MLIB_SHORT	Y	Y	Y	Y	Y	Y	MLIB_USHORT	Y	Y	Y	Y	Y	Y	MLIB_INT	Y	Y	Y	Y	Y	Y
Type [*]	BYTE	SHORT	USHORT	INT	FLOAT	DOUBLE																														
MLIB_BYTE	Y	Y	Y	Y	Y	Y																														
MLIB_SHORT	Y	Y	Y	Y	Y	Y																														
MLIB_USHORT	Y	Y	Y	Y	Y	Y																														
MLIB_INT	Y	Y	Y	Y	Y	Y																														
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>alpha</i> Scaling factor. <code>alpha[i]</code> contains the scaling factor for channel <i>i</i>.</p> <p><i>beta</i> Offset value. <code>beta[i]</code> contains the offset for channel <i>i</i>.</p> <p><i>shift</i> Right shifting factor.</p>																																			
RETURN VALUES	<p>The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code>.</p>																																			

mllib_ImageScale(3MLIB)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageScale_Fp(3MLIB)`, `mllib_ImageScale_Fp_Inp(3MLIB)`,
`mllib_ImageScale_Inp(3MLIB)`, `mllib_ImageScale2(3MLIB)`,
`mllib_ImageScale2_Inp(3MLIB)`, `attributes(5)`

mllib_ImageScale_Fp(3MLIB)

NAME	mllib_ImageScale_Fp – linear scaling																					
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageScale_Fp(mllib_image *dst, const mllib_image *src, const mllib_d64 *alpha, const mllib_d64 *beta);</pre>																					
DESCRIPTION	<p>The <code>mllib_ImageScale_Fp()</code> function performs a floating-point linear scaling on the pixels of the source image by multiplying the data by a scale factor, shifting, and then adding an offset.</p> <p>The following equation is used:</p> $dst[x][y][i] = src[x][y][i] * alpha[i] + beta[i]$ <p>If the result of the operation underflows/overflows the minimum/maximum value supported by the destination image, then it will be clamped to the minimum/maximum value respectively.</p> <p>See the following table for available variations of this linear scaling function.</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th>Type [*]</th> <th>BYTE</th> <th>SHORT</th> <th>USHORT</th> <th>INT</th> <th>FLOAT</th> <th>DOUBLE</th> </tr> </thead> <tbody> <tr> <td>MLIB_FLOAT</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> </tr> <tr> <td>MLIB_DOUBLE</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> <td>Y</td> </tr> </tbody> </table> <p>[*] Each row represents a source data type. Each column represents a destination data type.</p>	Type [*]	BYTE	SHORT	USHORT	INT	FLOAT	DOUBLE	MLIB_FLOAT	Y	Y	Y	Y	Y	Y	MLIB_DOUBLE	Y	Y	Y	Y	Y	Y
Type [*]	BYTE	SHORT	USHORT	INT	FLOAT	DOUBLE																
MLIB_FLOAT	Y	Y	Y	Y	Y	Y																
MLIB_DOUBLE	Y	Y	Y	Y	Y	Y																
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>alpha</i> Scaling factor. <code>alpha[i]</code> contains the scaling factor for channel <i>i</i>.</p> <p><i>beta</i> Offset value. <code>beta[i]</code> contains the offset for channel <i>i</i>.</p>																					
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .																					
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:																					
	<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe															
ATTRIBUTE TYPE	ATTRIBUTE VALUE																					
Interface Stability	Evolving																					
MT-Level	MT-Safe																					

`mlib_ImageScale_Fp(3MLIB)`

SEE ALSO `mlib_ImageScale(3MLIB)`, `mlib_ImageScale_Fp_Inp(3MLIB)`,
`mlib_ImageScale_Inp(3MLIB)`, `mlib_ImageScale2(3MLIB)`,
`mlib_ImageScale2_Inp(3MLIB)`, `attributes(5)`

mllib_ImageScale_Fp_Inp(3MLIB)

NAME	mllib_ImageScale_Fp_Inp – linear scaling, in place						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageScale_Fp_Inp(mllib_image *<i>srcdst</i>, const mllib_d64 *<i>alpha</i>, const mllib_d64 *<i>beta</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageScale_Fp_Inp()</code> function performs a floating-point, in-place linear scaling on the pixels of the source image by multiplying the data by a scale factor, shifting, and then adding an offset.</p> <p>The following equation is used:</p> $\text{srcdst}[x][y][i] = \text{srcdst}[x][y][i] * \text{alpha}[i] + \text{beta}[i]$ <p>If the result of the operation underflows/overflows the minimum/maximum value supported by the destination image, then it will be clamped to the minimum/maximum value respectively.</p> <p>The image can be of type <code>MLIB_FLOAT</code> or <code>MLIB_DOUBLE</code>.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>srcdst</i> Pointer to source and destination image.</p> <p><i>alpha</i> Scaling factor. <code>alpha[i]</code> contains the scaling factor for channel <i>i</i>.</p> <p><i>beta</i> Offset value. <code>beta[i]</code> contains the offset for channel <i>i</i>.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageScale(3MLIB)</code> , <code>mllib_ImageScale_Fp(3MLIB)</code> , <code>mllib_ImageScale_Inp(3MLIB)</code> , <code>mllib_ImageScale2(3MLIB)</code> , <code>mllib_ImageScale2_Inp(3MLIB)</code> , <code>attributes(5)</code>						

NAME	mllib_ImageScale_Inp – linear scaling, in place						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageScale_Inp(mllib_image *<i>srcdst</i>, const mllib_s32 *<i>alpha</i>, const mllib_s32 *<i>beta</i>, mllib_s32 <i>shift</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageScale_Inp()</code> function performs an in-place linear scaling on the pixels of the source image by multiplying the data by a scale factor, shifting, and then adding an offset.</p> <p>The following equation is used:</p> $\text{srcdst}[x][y][i] = \text{srcdst}[x][y][i] * \alpha[i] * 2^{**(-\text{shift})} + \text{beta}[i]$ <p>If the result of the operation underflows/overflows the minimum/maximum value supported by the destination image, then it will be clamped to the minimum/maximum value respectively.</p> <p>The image can be of type <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code> or <code>MLIB_INT</code>.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>srcdst</i> Pointer to source and destination image.</p> <p><i>alpha</i> Scaling factor. <code>alpha[i]</code> contains the scaling factor for channel <i>i</i>.</p> <p><i>beta</i> Offset value. <code>beta[i]</code> contains the offset for channel <i>i</i>.</p> <p><i>shift</i> Right shifting factor.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageScale(3MLIB)</code> , <code>mllib_ImageScale_Fp(3MLIB)</code> , <code>mllib_ImageScale_Fp_Inp(3MLIB)</code> , <code>mllib_ImageScale2(3MLIB)</code> , <code>mllib_ImageScale2_Inp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageSConv3x3(3MLIB)

NAME	mllib_ImageSConv3x3 – separable 3x3 convolution						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageSConv3x3(mllib_image *dst, const mllib_image *src, const mllib_s32 *hkernel, const mllib_s32 *vkernel, mllib_s32 scale, mllib_s32 cmask, mllib_edge edge);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageSConv3x3()</code> function performs a separable 3x3 convolution on the source image by using the user-supplied kernel.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sum_{p=-dm}^{m-1-dm} \sum_{q=-dn}^{n-1-dn} src[x+p][y+q][i] * h[p] * v[q] * 2^{**(-2*scale)}$ <p>where $m = 3$, $n = 3$, $dm = (m - 1)/2 = 1$, $dn = (n - 1)/2 = 1$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>hkernel</i> Pointer to the horizontal kernel.</p> <p><i>vkernel</i> Pointer to the vertical kernel.</p> <p><i>scale</i> Scaling factor.</p> <p><i>cmask</i> Channel mask to indicate the channels to be convolved, each bit of which represents a channel in the image. The channels corresponding to one bits are those to be processed. For a single-channel image, the channel mask is ignored.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SRC_EXTEND</pre>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						

mllib_ImageSConv3x3(3MLIB)

SEE ALSO mllib_ImageConv2x2(3MLIB), mllib_ImageConv2x2_Fp(3MLIB),
mllib_ImageConv2x2Index(3MLIB), mllib_ImageConv3x3(3MLIB),
mllib_ImageConv3x3_Fp(3MLIB), mllib_ImageConv3x3Index(3MLIB),
mllib_ImageConv4x4(3MLIB), mllib_ImageConv4x4_Fp(3MLIB),
mllib_ImageConv4x4Index(3MLIB), mllib_ImageConv5x5(3MLIB),
mllib_ImageConv5x5_Fp(3MLIB), mllib_ImageConv5x5Index(3MLIB),
mllib_ImageConv7x7(3MLIB), mllib_ImageConv7x7_Fp(3MLIB),
mllib_ImageConv7x7Index(3MLIB), mllib_ImageConvKernelConvert(3MLIB),
mllib_ImageConvMxN(3MLIB), mllib_ImageConvMxN_Fp(3MLIB),
mllib_ImageConvMxNIndex(3MLIB), mllib_ImageConvolveMxN(3MLIB),
mllib_ImageConvolveMxN_Fp(3MLIB), mllib_ImageSConv3x3_Fp(3MLIB),
mllib_ImageSConv5x5(3MLIB), mllib_ImageSConv5x5_Fp(3MLIB),
mllib_ImageSConv7x7(3MLIB), mllib_ImageSConv7x7_Fp(3MLIB),
mllib_ImageSConvKernelConvert(3MLIB), attributes(5)

mllib_ImageSConv3x3_Fp(3MLIB)

NAME	mllib_ImageSConv3x3_Fp – separable 3x3 convolution						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageSConv3x3_Fp(mllib_image *dst, const mllib_image *src, const mllib_d64 *hkernel, const mllib_d64 *vkernel, mllib_s32 cmask, mllib_edge edge);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageSConv3x3_Fp()</code> function performs a separable 3x3 convolution on the source image by using the user-supplied kernel.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sum_{p=-dm}^{m-1-dm} \sum_{q=-dn}^{n-1-dn} src[x+p][y+q][i] * h[p] * v[q]$ <p>where $m = 3$, $n = 3$, $dm = (m - 1) / 2 = 1$, $dn = (n - 1) / 2 = 1$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>hkernel</i> Pointer to the horizontal kernel.</p> <p><i>vkernel</i> Pointer to the vertical kernel.</p> <p><i>cmask</i> Channel mask to indicate the channels to be convolved, each bit of which represents a channel in the image. The channels corresponding to one bits are those to be processed. For a single-channel image, the channel mask is ignored.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SRC_EXTEND</pre>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						

mllib_ImageSConv3x3_Fp(3MLIB)

SEE ALSO

mllib_ImageConv2x2(3MLIB), mllib_ImageConv2x2_Fp(3MLIB),
mllib_ImageConv2x2Index(3MLIB), mllib_ImageConv3x3(3MLIB),
mllib_ImageConv3x3_Fp(3MLIB), mllib_ImageConv3x3Index(3MLIB),
mllib_ImageConv4x4(3MLIB), mllib_ImageConv4x4_Fp(3MLIB),
mllib_ImageConv4x4Index(3MLIB), mllib_ImageConv5x5(3MLIB),
mllib_ImageConv5x5_Fp(3MLIB), mllib_ImageConv5x5Index(3MLIB),
mllib_ImageConv7x7(3MLIB), mllib_ImageConv7x7_Fp(3MLIB),
mllib_ImageConv7x7Index(3MLIB), mllib_ImageConvKernelConvert(3MLIB),
mllib_ImageConvMxN(3MLIB), mllib_ImageConvMxN_Fp(3MLIB),
mllib_ImageConvMxNIndex(3MLIB), mllib_ImageConvolveMxN(3MLIB),
mllib_ImageConvolveMxN_Fp(3MLIB), mllib_ImageSConv3x3(3MLIB),
mllib_ImageSConv5x5(3MLIB), mllib_ImageSConv5x5_Fp(3MLIB),
mllib_ImageSConv7x7(3MLIB), mllib_ImageSConv7x7_Fp(3MLIB),
mllib_ImageSConvKernelConvert(3MLIB), attributes(5)

mllib_ImageSConv5x5(3MLIB)

NAME	mllib_ImageSConv5x5 – separable 5x5 convolution						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageSConv5x5(mllib_image *dst, const mllib_image *src, const mllib_s32 *hkernel, const mllib_s32 *vkernel, mllib_s32 scale, mllib_s32 cmask, mllib_edge edge);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageSConv5x5()</code> function performs a separable 5x5 convolution on the source image by using the user-supplied kernel.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sum_{p=-dm}^{m-1-dm} \sum_{q=-dn}^{n-1-dn} src[x+p][y+q][i] * h[p] * v[q] * 2^{**(-2*scale)}$ <p>where $m = 5$, $n = 5$, $dm = (m - 1) / 2 = 2$, $dn = (n - 1) / 2 = 2$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>hkernel</i> Pointer to the horizontal kernel.</p> <p><i>vkernel</i> Pointer to the vertical kernel.</p> <p><i>scale</i> Scaling factor.</p> <p><i>cmask</i> Channel mask to indicate the channels to be convolved, each bit of which represents a channel in the image. The channels corresponding to one bits are those to be processed. For a single-channel image, the channel mask is ignored.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SRC_EXTEND</pre>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						

mllib_ImageSConv5x5(3MLIB)

SEE ALSO mllib_ImageConv2x2(3MLIB), mllib_ImageConv2x2_Fp(3MLIB),
mllib_ImageConv2x2Index(3MLIB), mllib_ImageConv3x3(3MLIB),
mllib_ImageConv3x3_Fp(3MLIB), mllib_ImageConv3x3Index(3MLIB),
mllib_ImageConv4x4(3MLIB), mllib_ImageConv4x4_Fp(3MLIB),
mllib_ImageConv4x4Index(3MLIB), mllib_ImageConv5x5(3MLIB),
mllib_ImageConv5x5_Fp(3MLIB), mllib_ImageConv5x5Index(3MLIB),
mllib_ImageConv7x7(3MLIB), mllib_ImageConv7x7_Fp(3MLIB),
mllib_ImageConv7x7Index(3MLIB), mllib_ImageConvKernelConvert(3MLIB),
mllib_ImageConvMxN(3MLIB), mllib_ImageConvMxN_Fp(3MLIB),
mllib_ImageConvMxNIndex(3MLIB), mllib_ImageConvolveMxN(3MLIB),
mllib_ImageConvolveMxN_Fp(3MLIB), mllib_ImageSConv3x3(3MLIB),
mllib_ImageSConv3x3_Fp(3MLIB), mllib_ImageSConv5x5_Fp(3MLIB),
mllib_ImageSConv7x7(3MLIB), mllib_ImageSConv7x7_Fp(3MLIB),
mllib_ImageSConvKernelConvert(3MLIB), attributes(5)

mllib_ImageSConv5x5_Fp(3MLIB)

NAME	mllib_ImageSConv5x5_Fp – separable 5x5 convolution						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageSConv5x5_Fp(mllib_image *dst, const mllib_image *src, const mllib_d64 *hkernel, const mllib_d64 *vkernel, mllib_s32 cmask, mllib_edge edge);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageSConv5x5_Fp()</code> function performs a separable 5x5 convolution on the source image by using the user-supplied kernel.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sum_{p=-dm}^{m-1-dm} \sum_{q=-dn}^{n-1-dn} src[x+p][y+q][i] * h[p] * v[q]$ <p>where $m = 5$, $n = 5$, $dm = (m - 1) / 2 = 2$, $dn = (n - 1) / 2 = 2$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>hkernel</i> Pointer to the horizontal kernel.</p> <p><i>vkernel</i> Pointer to the vertical kernel.</p> <p><i>cmask</i> Channel mask to indicate the channels to be convolved, each bit of which represents a channel in the image. The channels corresponding to one bits are those to be processed. For a single-channel image, the channel mask is ignored.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SRC_EXTEND</pre>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						

mllib_ImageSConv5x5_Fp(3MLIB)

SEE ALSO

mllib_ImageConv2x2(3MLIB), mllib_ImageConv2x2_Fp(3MLIB),
mllib_ImageConv2x2Index(3MLIB), mllib_ImageConv3x3(3MLIB),
mllib_ImageConv3x3_Fp(3MLIB), mllib_ImageConv3x3Index(3MLIB),
mllib_ImageConv4x4(3MLIB), mllib_ImageConv4x4_Fp(3MLIB),
mllib_ImageConv4x4Index(3MLIB), mllib_ImageConv5x5(3MLIB),
mllib_ImageConv5x5_Fp(3MLIB), mllib_ImageConv5x5Index(3MLIB),
mllib_ImageConv7x7(3MLIB), mllib_ImageConv7x7_Fp(3MLIB),
mllib_ImageConv7x7Index(3MLIB), mllib_ImageConvKernelConvert(3MLIB),
mllib_ImageConvMxN(3MLIB), mllib_ImageConvMxN_Fp(3MLIB),
mllib_ImageConvMxNIndex(3MLIB), mllib_ImageConvolveMxN(3MLIB),
mllib_ImageConvolveMxN_Fp(3MLIB), mllib_ImageSConv3x3(3MLIB),
mllib_ImageSConv3x3_Fp(3MLIB), mllib_ImageSConv5x5(3MLIB),
mllib_ImageSConv7x7(3MLIB), mllib_ImageSConv7x7_Fp(3MLIB),
mllib_ImageSConvKernelConvert(3MLIB), attributes(5)

mllib_ImageSConv7x7(3MLIB)

NAME	mllib_ImageSConv7x7 – separable 7x7 convolution						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageSConv7x7(mllib_image *dst, const mllib_image *src, const mllib_s32 *hkernel, const mllib_s32 *vkernel, mllib_s32 scale, mllib_s32 cmask, mllib_edge edge);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageSConv7x7()</code> function performs a separable 7x7 convolution on the source image by using the user-supplied kernel.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sum_{p=-dm}^{m-1-dm} \sum_{q=-dn}^{n-1-dn} src[x+p][y+q][i] * h[p] * v[q] * 2^{**(-2*scale)}$ <p>where $m = 7$, $n = 7$, $dm = (m - 1)/2 = 3$, $dn = (n - 1)/2 = 3$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>hkernel</i> Pointer to the horizontal kernel.</p> <p><i>vkernel</i> Pointer to the vertical kernel.</p> <p><i>scale</i> Scaling factor.</p> <p><i>cmask</i> Channel mask to indicate the channels to be convolved, each bit of which represents a channel in the image. The channels corresponding to 1 bits are those to be processed. For a single-channel image, the channel mask is ignored.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SRC_EXTEND</pre>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						

`mlib_ImageSConv7x7(3MLIB)`

SEE ALSO `mlib_ImageConv2x2(3MLIB)`, `mlib_ImageConv2x2_Fp(3MLIB)`,
`mlib_ImageConv2x2Index(3MLIB)`, `mlib_ImageConv3x3(3MLIB)`,
`mlib_ImageConv3x3_Fp(3MLIB)`, `mlib_ImageConv3x3Index(3MLIB)`,
`mlib_ImageConv4x4(3MLIB)`, `mlib_ImageConv4x4_Fp(3MLIB)`,
`mlib_ImageConv4x4Index(3MLIB)`, `mlib_ImageConv5x5(3MLIB)`,
`mlib_ImageConv5x5_Fp(3MLIB)`, `mlib_ImageConv5x5Index(3MLIB)`,
`mlib_ImageConv7x7(3MLIB)`, `mlib_ImageConv7x7_Fp(3MLIB)`,
`mlib_ImageConv7x7Index(3MLIB)`, `mlib_ImageConvKernelConvert(3MLIB)`,
`mlib_ImageConvMxN(3MLIB)`, `mlib_ImageConvMxN_Fp(3MLIB)`,
`mlib_ImageConvMxNIndex(3MLIB)`, `mlib_ImageConvolveMxN(3MLIB)`,
`mlib_ImageConvolveMxN_Fp(3MLIB)`, `mlib_ImageSConv3x3(3MLIB)`,
`mlib_ImageSConv3x3_Fp(3MLIB)`, `mlib_ImageSConv5x5(3MLIB)`,
`mlib_ImageSConv5x5_Fp(3MLIB)`, `mlib_ImageSConv7x7_Fp(3MLIB)`,
`mlib_ImageSConvKernelConvert(3MLIB)`, `attributes(5)`

mllib_ImageSConv7x7_Fp(3MLIB)

NAME	mllib_ImageSConv7x7_Fp – separable 7x7 convolution						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageSConv7x7_Fp(mllib_image *dst, const mllib_image *src, const mllib_d64 *hkernel, const mllib_d64 *vkernel, mllib_s32 cmask, mllib_edge edge);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageSConv7x7_Fp()</code> function performs a separable 7x7 convolution on the source image by using the user-supplied kernel.</p> <p>It uses the following equation:</p> $dst[x][y][i] = \sum_{p=-dm}^{m-1-dm} \sum_{q=-dn}^{n-1-dn} src[x+p][y+q][i] * h[p] * v[q]$ <p>where $m = 7$, $n = 7$, $dm = (m - 1) / 2 = 3$, $dn = (n - 1) / 2 = 3$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>hkernel</i> Pointer to the horizontal kernel.</p> <p><i>vkernel</i> Pointer to the vertical kernel.</p> <p><i>cmask</i> Channel mask to indicate the channels to be convolved, each bit of which represents a channel in the image. The channels corresponding to 1 bits are those to be processed. For a single-channel image, the channel mask is ignored.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SRC_EXTEND</pre>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						

mllib_ImageSConv7x7_Fp(3MLIB)

SEE ALSO

mllib_ImageConv2x2(3MLIB), mllib_ImageConv2x2_Fp(3MLIB),
mllib_ImageConv2x2Index(3MLIB), mllib_ImageConv3x3(3MLIB),
mllib_ImageConv3x3_Fp(3MLIB), mllib_ImageConv3x3Index(3MLIB),
mllib_ImageConv4x4(3MLIB), mllib_ImageConv4x4_Fp(3MLIB),
mllib_ImageConv4x4Index(3MLIB), mllib_ImageConv5x5(3MLIB),
mllib_ImageConv5x5_Fp(3MLIB), mllib_ImageConv5x5Index(3MLIB),
mllib_ImageConv7x7(3MLIB), mllib_ImageConv7x7_Fp(3MLIB),
mllib_ImageConv7x7Index(3MLIB), mllib_ImageConvKernelConvert(3MLIB),
mllib_ImageConvMxN(3MLIB), mllib_ImageConvMxN_Fp(3MLIB),
mllib_ImageConvMxNIndex(3MLIB), mllib_ImageConvolveMxN(3MLIB),
mllib_ImageConvolveMxN_Fp(3MLIB), mllib_ImageSConv3x3(3MLIB),
mllib_ImageSConv3x3_Fp(3MLIB), mllib_ImageSConv5x5(3MLIB),
mllib_ImageSConv5x5_Fp(3MLIB), mllib_ImageSConv7x7(3MLIB),
mllib_ImageSConvKernelConvert(3MLIB), attributes(5)

mllib_ImageSConvKernelConvert(3MLIB)

NAME	mllib_ImageSConvKernelConvert – kernel conversion for separable convolution																
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageSConvKernelConvert (mllib_s32 *<i>ihkernel</i>, mllib_s32 *<i>ivkernel</i>, mllib_s32 *<i>iscale</i>, const mllib_d64 *<i>fhkernel</i>, const mllib_d64 *<i>fvkernel</i>, mllib_s32 <i>m</i>, mllib_s32 <i>n</i>, mllib_type <i>type</i>);</pre>																
DESCRIPTION	The <code>mllib_ImageSConvKernelConvert()</code> function converts a floating-point separable convolution kernel to an integer kernel with its scaling factor, which is suitable to be used in separable convolution functions.																
PARAMETERS	The function takes the following arguments: <table><tr><td><i>ihkernel</i></td><td>Pointer to integer horizontal kernel.</td></tr><tr><td><i>ivkernel</i></td><td>Pointer to integer vertical kernel.</td></tr><tr><td><i>iscale</i></td><td>Scaling factor of the integer convolution kernel.</td></tr><tr><td><i>fhkernel</i></td><td>Pointer to floating-point horizontal kernel.</td></tr><tr><td><i>fvkernel</i></td><td>Pointer to floating-point vertical kernel.</td></tr><tr><td><i>m</i></td><td>Width of the convolution kernel. <i>m</i> must be an odd number larger than 1.</td></tr><tr><td><i>n</i></td><td>Height of the convolution kernel. <i>n</i> must be an odd number larger than 1.</td></tr><tr><td><i>type</i></td><td>The image type.</td></tr></table>	<i>ihkernel</i>	Pointer to integer horizontal kernel.	<i>ivkernel</i>	Pointer to integer vertical kernel.	<i>iscale</i>	Scaling factor of the integer convolution kernel.	<i>fhkernel</i>	Pointer to floating-point horizontal kernel.	<i>fvkernel</i>	Pointer to floating-point vertical kernel.	<i>m</i>	Width of the convolution kernel. <i>m</i> must be an odd number larger than 1.	<i>n</i>	Height of the convolution kernel. <i>n</i> must be an odd number larger than 1.	<i>type</i>	The image type.
<i>ihkernel</i>	Pointer to integer horizontal kernel.																
<i>ivkernel</i>	Pointer to integer vertical kernel.																
<i>iscale</i>	Scaling factor of the integer convolution kernel.																
<i>fhkernel</i>	Pointer to floating-point horizontal kernel.																
<i>fvkernel</i>	Pointer to floating-point vertical kernel.																
<i>m</i>	Width of the convolution kernel. <i>m</i> must be an odd number larger than 1.																
<i>n</i>	Height of the convolution kernel. <i>n</i> must be an odd number larger than 1.																
<i>type</i>	The image type.																
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .																
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe										
ATTRIBUTE TYPE	ATTRIBUTE VALUE																
Interface Stability	Evolving																
MT-Level	MT-Safe																
SEE ALSO	<code>mllib_ImageSConv3x3(3MLIB)</code> , <code>mllib_ImageSConv3x3_Fp(3MLIB)</code> , <code>mllib_ImageSConv5x5(3MLIB)</code> , <code>mllib_ImageSConv5x5_Fp(3MLIB)</code> , <code>mllib_ImageSConv7x7(3MLIB)</code> , <code>mllib_ImageSConv7x7_Fp(3MLIB)</code> , <code>mllib_ImageConvKernelConvert(3MLIB)</code> , <code>attributes(5)</code>																

NAME mllib_ImageSetFormat – set format

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageSetFormat(mllib_image *img, mllib_format
    format);
```

DESCRIPTION The mllib_ImageSetFormat() function sets a new value for the *format* field of a mllib_image structure.

The data type of the image can be MLIB_BIT, MLIB_BYTE, MLIB_SHORT, MLIB_USHORT, MLIB_INT, MLIB_FLOAT, or MLIB_DOUBLE.

PARAMETERS The function takes the following arguments:

img Pointer to a mediaLib image structure.

format Image pixel format. It can be one of the following:

```
MLIB_FORMAT_UNKNOWN
MLIB_FORMAT_INDEXED
MLIB_FORMAT_GRAYSCALE
MLIB_FORMAT_RGB
MLIB_FORMAT_BGR
MLIB_FORMAT_ARGB
MLIB_FORMAT_ABGR
MLIB_FORMAT_PACKED_ARGB
MLIB_FORMAT_PACKED_ABGR
MLIB_FORMAT_GRAYSCALE_ALPHA
MLIB_FORMAT_RGBA
```

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageGetFormat(3MLIB), mllib_ImageCreate(3MLIB), mllib_ImageCreateStruct(3MLIB), mllib_ImageCreateSubimage(3MLIB), attributes(5)

mllib_ImageSetPaddings(3MLIB)

NAME	mllib_ImageSetPaddings – set paddings										
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageSetPaddings(mllib_image *<i>img</i>, mllib_u8 <i>left</i>, mllib_u8 <i>top</i>, mllib_u8 <i>right</i>, mllib_u8 <i>bottom</i>);</pre>										
DESCRIPTION	<p>The <code>mllib_ImageSetPaddings()</code> function sets new values for the <code>paddings</code> field of the <code>mllib_image</code> structure as follows:</p> <pre>img->paddings[0] = left; img->paddings[1] = top; img->paddings[2] = right; img->paddings[3] = bottom;</pre> <p>By default, an image structure creation function, such as <code>mllib_ImageCreate()</code>, <code>mllib_ImageCreateStruct()</code>, or <code>mllib_ImageCreateSubimage()</code>, sets the <code>paddings</code> field of the <code>mllib_image</code> structure as follows:</p> <pre>img->paddings[0] = 0; img->paddings[1] = 0; img->paddings[2] = 0; img->paddings[3] = 0;</pre> <p>Note that this function is needed only when the edge condition <code>MLIB_EDGE_SRC_PADDED</code> is used.</p> <p>The <code>mllib_image->paddings</code> field denotes the amount of paddings on each side of an image, from which the real image border can be seen. When <code>MLIB_EDGE_SRC_PADDED</code> is specified as the edge condition, a geometric function uses the "real" source image border for clipping the destination image.</p>										
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>img</i></td><td>Pointer to image data structure.</td></tr><tr><td><i>left</i></td><td>Number of columns padded on the left side.</td></tr><tr><td><i>top</i></td><td>Number of rows padded on the top.</td></tr><tr><td><i>right</i></td><td>Number of columns padded on the right side.</td></tr><tr><td><i>bottom</i></td><td>Number of rows padded at the bottom.</td></tr></table>	<i>img</i>	Pointer to image data structure.	<i>left</i>	Number of columns padded on the left side.	<i>top</i>	Number of rows padded on the top.	<i>right</i>	Number of columns padded on the right side.	<i>bottom</i>	Number of rows padded at the bottom.
<i>img</i>	Pointer to image data structure.										
<i>left</i>	Number of columns padded on the left side.										
<i>top</i>	Number of rows padded on the top.										
<i>right</i>	Number of columns padded on the right side.										
<i>bottom</i>	Number of rows padded at the bottom.										
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .										
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:										
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving						
ATTRIBUTE TYPE	ATTRIBUTE VALUE										
Interface Stability	Evolving										

mllib_ImageSetPaddings(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	MT-Safe

SEE ALSO `mllib_ImageGetPaddings(3MLIB)`, `mllib_ImageCreate(3MLIB)`,
`mllib_ImageCreateStruct(3MLIB)`, `mllib_ImageCreateSubimage(3MLIB)`,
`mllib_ImageAffine(3MLIB)`, `attributes(5)`

mllib_ImageSobel(3MLIB)

NAME	mllib_ImageSobel, mllib_ImageSobel_Fp – Sobel filter						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageSobel(mllib_image *dst, const mllib_image *src, mllib_s32 cmask, mllib_edge edge); mllib_status mllib_ImageSobel_Fp(mllib_image *dst, const mllib_image *src, mllib_s32 cmask, mllib_edge edge);</pre>						
DESCRIPTION	<p>Each function is a special case of the gradient filter, which is an edge detector which computes the magnitude of the image gradient vector in two orthogonal directions. In this case, the gradient filter uses specific horizontal and vertical masks.</p> <p>The Sobel filter is one of the special cases of gradient filter using the following horizontal and vertical masks:</p> <pre>hmask = { -1.0, 0.0, 1.0, -2.0, 0.0, 2.0, -1.0, 0.0, 1.0 }</pre> <pre>vmask = { -1.0, -2.0, -1.0, 0.0, 0.0, 0.0, 1.0, 2.0, 1.0 }</pre>						
PARAMETERS	<p>Each function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>cmask</i> Channel mask to indicate the channels to be processed, each bit of which represents a channel in the image. The channels corresponding to 1 bits are those to be processed. For a single channel image, the channel mask is ignored.</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DS_FILL_ZERO MLIB_EDGE_DST_COPY_SRC MLIB_EDGE_SR_EXTEND</pre>						
RETURN VALUES	The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						

`mllib_ImageSobel(3MLIB)`

SEE ALSO `mllib_ImageGradient3x3(3MLIB)`, `mllib_ImageGradient3x3_Fp(3MLIB)`,
`attributes(5)`

mllib_ImageSqr_Fp(3MLIB)

NAME	mllib_ImageSqr_Fp – square						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageSqr_Fp(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageSqr_Fp()</code> function computes the floating-point square of each pixel in the source image.</p> <p>It uses the following equation:</p> $dst[x][y][i] = src[x][y][i] * src[x][y][i]$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageSqr_Fp_Inp(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageSqr_Fp_Inp – square, in place

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageSqr_Fp_Inp(mllib_image *srcdst);
```

DESCRIPTION The `mllib_ImageSqr_Fp_Inp()` function computes the floating-point square of each pixel in the source image.

It uses the following equation:

$$\text{srcdst}[x][y][i] = \text{srcdst}[x][y][i] * \text{srcdst}[x][y][i]$$

PARAMETERS The function takes the following arguments:

srcdst Pointer to source and destination image.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_ImageSqr_Fp\(3MLIB\)](#), [attributes\(5\)](#)

mllib_ImageSqrShift(3MLIB)

NAME | mllib_ImageSqrShift – square with shifting

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageSqrShift(mllib_image *dst, const mllib_image
    *src, mllib_s32 shift);
```

DESCRIPTION | The `mllib_ImageSqrShift()` function computes the square of each pixel in the source image and scales the result by the shift factor.

It uses the following equation:

$$dst[x][y][i] = src[x][y][i] * src[x][y][i] * 2^{*(-shift)}$$

PARAMETERS | The function takes the following arguments:

dst | Pointer to destination image.

src | Pointer to source image.

shift | Right shifting factor. $0 \leq shift \leq 31$.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_ImageSqrShift_Inp\(3MLIB\)](#), [attributes\(5\)](#)

NAME mllib_ImageSqrShift_Inp – square with shifting, in place

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageSqrShift_Inp(mllib_image *srcdst, mllib_s32
    shift);
```

DESCRIPTION The `mllib_ImageSqrShift_Inp()` function computes the square of each pixel in the source image and scales the result by the shift factor, in place.

It uses the following equation:

$$\text{srcdst}[x][y][i] = \text{srcdst}[x][y][i] * \text{srcdst}[x][y][i] * 2^{**(-\text{shift})}$$

PARAMETERS The function takes the following arguments:

srcdst Pointer to source and destination image.

shift Right shifting factor. $0 \leq \text{shift} \leq 31$.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageSqrShift(3MLIB)`, `attributes(5)`

mllib_ImageStdDev(3MLIB)

NAME mllib_ImageStdDev – image standard deviation

SYNOPSIS cc [*flag...*] *file...* -lmllib [*library...*]
#include <mllib.h>

```
mllib_status mllib_ImageStdDev(mllib_d64 *sdev, const mllib_image
    *img, const mllib_d64 *mean);
```

DESCRIPTION The mllib_ImageStdDev() function computes the standard deviation for each channel in the source image.

It uses the following equation:

$$sdev[i] = \left\{ \frac{1}{w \cdot h} * \sum_{x=0}^{w-1} \sum_{y=0}^{h-1} (img[x][y][i] - mean[i])**2 \right\} ** 0.5$$

where, in the case of mean == NULL,

$$mean[i] = \frac{1}{w \cdot h} * \sum_{x=0}^{w-1} \sum_{y=0}^{h-1} img[x][y][i]$$

PARAMETERS The function takes the following arguments:

sdev Pointer to standard deviation array, whose size is the number of channels in the source image. sdev[i] contains the standard deviation of channel i.

img Pointer to input image.

mean Pointer to pre-computed mean array for each channel. (If NULL, it will be computed.) mean[i] contains the mean of channel i.

RETURN VALUES The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageMean(3MLIB), mllib_ImageMean_Fp(3MLIB), mllib_ImageStdDev_Fp(3MLIB), attributes(5)

NAME mllib_ImageStdDev_Fp – image standard deviation

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageStdDev_Fp(mllib_d64 *sdev, const mllib_image
    *img, const mllib_d64 *mean);
```

DESCRIPTION The mllib_ImageStdDev_Fp() function computes the standard deviation for each channel in the floating-point source image.

It uses the following equation:

$$sdev[i] = \left\{ \frac{1}{w \cdot h} * \sum_{x=0}^{w-1} \sum_{y=0}^{h-1} (img[x][y][i] - mean[i])**2 \right\} ** 0.5$$

where, in the case of mean == NULL,

$$mean[i] = \frac{1}{w \cdot h} * \sum_{x=0}^{w-1} \sum_{y=0}^{h-1} img[x][y][i]$$

PARAMETERS The function takes the following arguments:

sdev Pointer to standard deviation array, whose size is the number of channels in the source image. *sdev[i]* contains the standard deviation of channel *i*.

img Pointer to input image.

mean Pointer to pre-computed mean array for each channel. (If NULL, it will be computed.) *mean[i]* contains the mean of channel *i*.

RETURN VALUES The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageMean(3MLIB), mllib_ImageMean_Fp(3MLIB), mllib_ImageStdDev(3MLIB), attributes(5)

mllib_ImageSub1_Fp_Inp(3MLIB)

NAME	mllib_ImageSub1_Fp_Inp – subtraction, in place						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageSub1_Fp_Inp(mllib_image *src1dst, const mllib_image *src2);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageSub1_Fp_Inp()</code> function subtracts the second floating-point source image from the first floating-point source image on a pixel-by-pixel basis, in place.</p> <p>It uses the following equation:</p> $\text{src1dst}[x][y][i] = \text{src1dst}[x][y][i] - \text{src2}[x][y][i]$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>src1dst</i> Pointer to first source and destination image.</p> <p><i>src2</i> Pointer to second source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageSub(3MLIB)</code> , <code>mllib_ImageSub_Fp(3MLIB)</code> , <code>mllib_ImageSub1_Inp(3MLIB)</code> , <code>mllib_ImageSub2_Fp_Inp(3MLIB)</code> , <code>mllib_ImageSub2_Inp(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageSub1_Inp – subtraction, in place

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageSub1_Inp(mllib_image *src1dst, const
    mllib_image *src2);
```

DESCRIPTION The mllib_ImageSub1_Inp() function subtracts the second source image from the first source image on a pixel-by-pixel basis, in place.

It uses the following equation:

$$src1dst[x][y][i] = src1dst[x][y][i] - src2[x][y][i]$$

PARAMETERS The function takes the following arguments:

src1dst Pointer to first source and destination image.

src2 Pointer to second source image.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageSub(3MLIB), mllib_ImageSub_Fp(3MLIB), mllib_ImageSub1_Fp_Inp(3MLIB), mllib_ImageSub2_Fp_Inp(3MLIB), mllib_ImageSub2_Inp(3MLIB), attributes(5)

mllib_ImageSub2_Fp_Inp(3MLIB)

NAME | mllib_ImageSub2_Fp_Inp – subtraction, in place

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageSub2_Fp_Inp(mllib_image *src2dst, const
mllib_image *src1);
```

DESCRIPTION | The `mllib_ImageSub2_Fp_Inp()` function subtracts the second floating-point source image from the first floating-point source image on a pixel-by-pixel basis, in place.

It uses the following equation:

$$\text{src2dst}[x][y][i] = \text{src1}[x][y][i] - \text{src2dst}[x][y][i]$$

PARAMETERS | The function takes the following arguments:

src2dst Pointer to second source and destination image.

src1 Pointer to first source image.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageSub(3MLIB)`, `mllib_ImageSub_Fp(3MLIB)`,
`mllib_ImageSub1_Fp_Inp(3MLIB)`, `mllib_ImageSub1_Inp(3MLIB)`,
`mllib_ImageSub2_Inp(3MLIB)`, `attributes(5)`

NAME | mllib_ImageSub2_Inp – subtraction, in place

SYNOPSIS | `cc [flag...] file... -lmllib [library...]
#include <mllib.h>`

`mllib_status mllib_ImageSub2_Inp(mllib_image *src2dst, const
mllib_image *src1);`

DESCRIPTION | The mllib_ImageSub2_Inp() function subtracts the second source image from the first source image on a pixel-by-pixel basis, in place.

It uses the following equation:

`src2dst[x][y][i] = src1[x][y][i] - src2dst[x][y][i]`

PARAMETERS | The function takes the following arguments:

`src2dst` | Pointer to second source and destination image.

`src1` | Pointer to first source image.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_ImageSub(3MLIB), mllib_ImageSub_Fp(3MLIB),
mllib_ImageSub1_Fp_Inp(3MLIB), mllib_ImageSub1_Inp(3MLIB),
mllib_ImageSub2_Fp_Inp(3MLIB), attributes(5)

mllib_ImageSub(3MLIB)

NAME	mllib_ImageSub – subtraction						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageSub(mllib_image *<i>dst</i>, const mllib_image *<i>src1</i>, const mllib_image *<i>src2</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageSub()</code> function subtracts the second source image from the first source image on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $dst[x][y][i] = src1[x][y][i] - src2[x][y][i]$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src1</i> Pointer to first source image.</p> <p><i>src2</i> Pointer to second source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageSub_Fp(3MLIB)</code> , <code>mllib_ImageSub1_Fp_Inp(3MLIB)</code> , <code>mllib_ImageSub1_Inp(3MLIB)</code> , <code>mllib_ImageSub2_Fp_Inp(3MLIB)</code> , <code>mllib_ImageSub2_Inp(3MLIB)</code> , <code>attributes(5)</code>						

NAME	mllib_ImageSub_Fp – subtraction
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageSub_Fp(mllib_image *dst, const mllib_image *src1, const mllib_image *src2);</pre>
DESCRIPTION	<p>The <code>mllib_ImageSub_Fp()</code> function subtracts the second floating-point source image from the first floating-point source image on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $dst[x][y][i] = src1[x][y][i] - src2[x][y][i]$
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src1</i> Pointer to first source image.</p> <p><i>src2</i> Pointer to second source image.</p>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageSub(3MLIB)`, `mllib_ImageSub1_Fp_Inp(3MLIB)`, `mllib_ImageSub1_Inp(3MLIB)`, `mllib_ImageSub2_Fp_Inp(3MLIB)`, `mllib_ImageSub2_Inp(3MLIB)`, `attributes(5)`

mllib_ImageSubsampleAverage(3MLIB)

NAME	mllib_ImageSubsampleAverage, mllib_ImageSubsampleAverage_Fp – subsamples an image with a box filter								
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageSubsampleAverage(mllib_image *dst, const mllib_image *src, mllib_d64 xscale, mllib_d64 yscale); mllib_status mllib_ImageSubsampleAverage_Fp(mllib_image *dst, const mllib_image *src, mllib_d64 xscale, mllib_d64 yscale);</pre>								
DESCRIPTION	<p>Each function scales an image down with an adaptive box filter.</p> <p>The subsampling algorithm performs the scaling operation by averaging all the pixel values from a block in the source image that correspond to the destination pixel.</p> <p>The width and height of the source block for a destination pixel are computed as:</p> <pre>blockX = (int)ceil(1.0/xscale); blockY = (int)ceil(1.0/yscale);</pre> <p>If we denote a pixel's location in an image by its column number and row number (both counted from 0), the destination pixel at (i, j) is backward mapped to the source block whose upper-left corner pixel is at (xValues[i], yValues[j]), where</p> <pre>xValues[i] = (int)(i/xscale + 0.5); yValues[j] = (int)(j/yscale + 0.5);</pre> <p>The width and height of the filled area in the destination are restricted by</p> <pre>dstW = (int)(srcWidth * xscale); dstH = (int)(srcHeight * yscale);</pre> <p>where srcWidth and srcHeight are width and height of the source image.</p> <p>Since the block size in source is defined from scale factors with roundup, some blocks (the rightmost and the bottommost blocks) may overrun the border of the source image by 1 pixel. In this case, such blocks are moved by 1 pixel to left/up direction in order to be inside of the source image.</p>								
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>dst</i></td><td>Pointer to destination image.</td></tr><tr><td><i>src</i></td><td>Pointer to source image.</td></tr><tr><td><i>xscale</i></td><td>X scale factor. $0.0 < xscale \leq 1.0$.</td></tr><tr><td><i>yscale</i></td><td>Y scale factor. $0.0 < yscale \leq 1.0$.</td></tr></table>	<i>dst</i>	Pointer to destination image.	<i>src</i>	Pointer to source image.	<i>xscale</i>	X scale factor. $0.0 < xscale \leq 1.0$.	<i>yscale</i>	Y scale factor. $0.0 < yscale \leq 1.0$.
<i>dst</i>	Pointer to destination image.								
<i>src</i>	Pointer to source image.								
<i>xscale</i>	X scale factor. $0.0 < xscale \leq 1.0$.								
<i>yscale</i>	Y scale factor. $0.0 < yscale \leq 1.0$.								
RETURN VALUES	The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.								

`mllib_ImageSubsampleAverage(3MLIB)`

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageSubsampleBinaryToGray(3MLIB)`,
`mllib_ImageFilteredSubsample(3MLIB)`, `mllib_ImageZoomTranslate(3MLIB)`,
`mllib_ImageZoomTranslate_Fp(3MLIB)`, `attributes(5)`

mllib_ImageSubsampleBinaryToGray(3MLIB)

NAME	mllib_ImageSubsampleBinaryToGray – subsamples a binary image and converts it to a grayscale image
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageSubsampleBinaryToGray(mllib_image *dst, const mllib_image *src, mllib_d64 xscale, mllib_d64 yscale, const mllib_u8 *lutGray);</pre>
DESCRIPTION	<p>The <code>mllib_ImageSubsampleBinaryToGray()</code> function subsamples a binary (MLIB_BIT) image and converts it to a grayscale (MLIB_BYTE) image.</p> <p>The subsampling algorithm performs the scaling operation by accumulating all the bits in the source image that correspond to the destination pixel and, based on the x and y scaling factors, reserving consecutive indexes in the colormap for the maximum number of gray levels possible in the destination image. The destination image pixel values of this function are either gray levels or indexes (if <code>lutGray==NULL</code>).</p> <p>For representing the source block of pixels that is used to determine destination pixel values, the index 0 represents a block with no 1's (all 0's), the index 1 represents a block with a single 1, and so on. If the scaling factors require a fractional block of source pixels to determine a destination pixel value, the block size is rounded up. For example, if a 2.2-by-2.2 block of source pixels would be required to determine destination pixel values, a 3-by-3 block is used, resulting in 10 possible gray levels and therefore 10 colormap indexes, whose values are 0 through 9.</p> <p>The width and height of the source block for a destination pixel are computed as:</p> <pre>blockX = (int)ceil(1.0/xscale); blockY = (int)ceil(1.0/yscale);</pre> <p>If we denote a pixel's location in an image by its column number and row number (both counted from 0), the destination pixel at (i, j) is backward mapped to the source block whose upper-left corner pixel is at (xValues[i], yValues[j]), where</p> <pre>xValues[i] = (int)(i/xscale + 0.5); yValues[j] = (int)(j/yscale + 0.5);</pre> <p>The width and height of the filled area in the destination are restricted by</p> <pre>dstW = (int)(srcWidth * xscale); dstH = (int)(srcHeight * yscale);</pre> <p>where <code>srcWidth</code> and <code>srcHeight</code> are width and height of the source image.</p> <p>Since the block size in source is defined from scale factors with roundup, some blocks (the rightmost and the bottommost blocks) may overrun the border of the source image by 1 pixel. In this case, such blocks are moved by 1 pixel to left/up direction in order to be inside of the source image.</p>
PARAMETERS	The function takes the following arguments:

mllib_ImageSubsampleBinaryToGray(3MLIB)

dst Pointer to destination image . It must be of type `MLIB_BYTE` and have just one channel.

src Pointer to source image. It must be of type `MLIB_BIT` and have just one channel.

xscale X scale factor. $0.0 < xscale \leq 1.0$.

yscale Y scale factor. $0.0 < yscale \leq 1.0$.

lutGray Pointer to a grayscale lookup-table.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageZoomTranslateToGray(3MLIB)`,
`mllib_ImageSubsampleAverage(3MLIB)`, `attributes(5)`

mllib_ImageTestFlags(3MLIB)

NAME	mllib_ImageTestFlags – test flags						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> int mllib_ImageTestFlags(const mllib_image *<i>img</i>, mllib_s32 <i>flags</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageTestFlags()</code> function tests the flags for a combination of the following predefined characteristics. Note that the result of zero means the conditions are satisfied.</p> <pre>MLIB_IMAGE_ALIGNED64 /* data address is 64-byte aligned */ MLIB_IMAGE_ALIGNED8 /* data address is 8-byte aligned */ MLIB_IMAGE_ALIGNED4 /* data address is 4-byte aligned */ MLIB_IMAGE_ALIGNED2 /* data address is 2-byte aligned */ MLIB_IMAGE_WIDTH8X /* width is multiple of 8 */ MLIB_IMAGE_WIDTH4X /* width is multiple of 4 */ MLIB_IMAGE_WIDTH2X /* width is multiple of 2 */ MLIB_IMAGE_HEIGHT8X /* height is multiple of 8 */ MLIB_IMAGE_HEIGHT4X /* height is multiple of 4 */ MLIB_IMAGE_HEIGHT2X /* height is multiple of 2 */ MLIB_IMAGE_STRIDE8X /* stride is multiple of 8 */ MLIB_IMAGE_ONEVECTOR /* stride is equal to width in bytes */ MLIB_IMAGE_USERALLOCATED /* data space has been allocated by user */ MLIB_IMAGE_ATTRIBUTESSET /* image attribute flags have been set */</pre>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>img</i> Pointer to a mediaLib image structure.</p> <p><i>flags</i> Combination of a set of characteristics to be tested. It is formed by logically Oring one or more individual predefined characteristics.</p>						
RETURN VALUES	The function returns an integer value containing results of test. Condition = 0 if satisfied; otherwise, Condition != 0.						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageGetFlags(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageThresh1 – image thresholding

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageThresh1(mllib_image *dst, const mllib_image
    *src, const mllib_s32 *thresh, const mllib_s32 *ghigh, const
    mllib_s32 *glow);
```

DESCRIPTION The mllib_ImageThresh1() function compares each pixel in the source image to a threshold value. If the pixel is less than or equal to the threshold value, then the destination pixel is set to the low output level. If the pixel is greater than the threshold value, then the destination pixel is set to the high output level.

The data type of the destination image can be MLIB_BIT or can be the same as the data type of the source image.

It uses the following equation:

```
dst[x][y][i] = glow[i]    if src[x][y][i] ≤ thresh[i]
dst[x][y][i] = ghigh[i]  if src[x][y][i] > thresh[i]
```

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

thresh Threshold value. thresh[i] contains the threshold for channel i.

ghigh High output level. ghigh[i] contains the high output level for channel i.

glow Low output level. glow[i] contains the low output level for channel i.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageThresh1_Fp(3MLIB), mllib_ImageThresh1_Fp_Inp(3MLIB), mllib_ImageThresh1_Inp(3MLIB), mllib_ImageThresh2(3MLIB), mllib_ImageThresh2_Fp(3MLIB), mllib_ImageThresh2_Fp_Inp(3MLIB), mllib_ImageThresh2_Inp(3MLIB), mllib_ImageThresh3(3MLIB), mllib_ImageThresh3_Fp(3MLIB), mllib_ImageThresh3_Fp_Inp(3MLIB), mllib_ImageThresh3_Inp(3MLIB), mllib_ImageThresh4(3MLIB), mllib_ImageThresh4_Fp(3MLIB), mllib_ImageThresh4_Fp_Inp(3MLIB),

`mlib_ImageThresh1(3MLIB)`

```
mllib_ImageThresh4_Inp(3MLIB), mllib_ImageThresh5(3MLIB),  
mllib_ImageThresh5_Fp(3MLIB), mllib_ImageThresh5_Fp_Inp(3MLIB),  
mllib_ImageThresh5_Inp(3MLIB), attributes(5)
```

NAME mllib_ImageThresh1_Fp – image thresholding

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageThresh1_Fp(mllib_image *dst, const
    mllib_image *src, const mllib_d64 *thresh, const mllib_d64 *ghigh,
    const mllib_d64 *glow);
```

DESCRIPTION The mllib_ImageThresh1_Fp() function compares each pixel in the floating-point source image to a threshold value. If the pixel is less than or equal to the threshold value, then the destination pixel is set to the low output level. If the pixel is greater than the threshold value, then the destination pixel is set to the high output level.

The data type of the destination image can be MLIB_BIT or can be the same as the data type of the source image.

It uses the following equation:

```
dst[x][y][i] = glow[i]    if src[x][y][i] ≤ thresh[i]
dst[x][y][i] = ghigh[i]  if src[x][y][i] > thresh[i]
```

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

thresh Threshold value. thresh[i] contains the threshold for channel i.

ghigh High output level. ghigh[i] contains the high output level for channel i.

glow Low output level. glow[i] contains the low output level for channel i.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageThresh1(3MLIB), mllib_ImageThresh1_Fp_Inp(3MLIB), mllib_ImageThresh1_Inp(3MLIB), mllib_ImageThresh2(3MLIB), mllib_ImageThresh2_Fp(3MLIB), mllib_ImageThresh2_Fp_Inp(3MLIB), mllib_ImageThresh2_Inp(3MLIB), mllib_ImageThresh3(3MLIB), mllib_ImageThresh3_Fp(3MLIB), mllib_ImageThresh3_Fp_Inp(3MLIB), mllib_ImageThresh3_Inp(3MLIB), mllib_ImageThresh4(3MLIB), mllib_ImageThresh4_Fp(3MLIB), mllib_ImageThresh4_Fp_Inp(3MLIB),

`mllib_ImageThresh1_Fp(3MLIB)`

```
mllib_ImageThresh4_Inp(3MLIB), mllib_ImageThresh5(3MLIB),  
mllib_ImageThresh5_Fp(3MLIB), mllib_ImageThresh5_Fp_Inp(3MLIB),  
mllib_ImageThresh5_Inp(3MLIB), attributes(5)
```

NAME mllib_ImageThresh1_Fp_Inp – image thresholding

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageThresh1_Fp_Inp(mllib_image *srcdst, const
    mllib_d64 *thresh, const mllib_d64 *ghigh, const mllib_d64 *glow);
```

DESCRIPTION The mllib_ImageThresh1_Fp_Inp() function compares each pixel in the floating-point source image to a threshold value, in place. If the pixel is less than or equal to the threshold value, then the destination pixel is set to the low output level. If the pixel is greater than the threshold value, then the destination pixel is set to the high output level.

It uses the following equation:

```
srcdst[x][y][i] = glow[i]    if srcdst[x][y][i] ≤ thresh[i]
srcdst[x][y][i] = ghigh[i]  if srcdst[x][y][i] > thresh[i]
```

PARAMETERS The function takes the following arguments:

srcdst Pointer to source and destination image.

thresh Threshold value. *thresh*[*i*] contains the threshold for channel *i*.

ghigh High output level. *ghigh*[*i*] contains the high output level for channel *i*.

glow Low output level. *glow*[*i*] contains the low output level for channel *i*.

RETURN VALUES The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageThresh1(3MLIB), mllib_ImageThresh1_Fp(3MLIB), mllib_ImageThresh1_Inp(3MLIB), mllib_ImageThresh2(3MLIB), mllib_ImageThresh2_Fp(3MLIB), mllib_ImageThresh2_Fp_Inp(3MLIB), mllib_ImageThresh2_Inp(3MLIB), mllib_ImageThresh3(3MLIB), mllib_ImageThresh3_Fp(3MLIB), mllib_ImageThresh3_Fp_Inp(3MLIB), mllib_ImageThresh3_Inp(3MLIB), mllib_ImageThresh4(3MLIB), mllib_ImageThresh4_Fp(3MLIB), mllib_ImageThresh4_Fp_Inp(3MLIB), mllib_ImageThresh4_Inp(3MLIB), mllib_ImageThresh5(3MLIB), mllib_ImageThresh5_Fp(3MLIB), mllib_ImageThresh5_Fp_Inp(3MLIB), mllib_ImageThresh5_Inp(3MLIB), attributes(5)

mllib_ImageThresh1_Inp(3MLIB)

NAME	mllib_ImageThresh1_Inp – image thresholding								
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageThresh1_Inp(mllib_image *<i>srcdst</i>, const mllib_s32 *<i>thresh</i>, const mllib_s32 *<i>ghigh</i>, const mllib_s32 *<i>glow</i>);</pre>								
DESCRIPTION	<p>The <code>mllib_ImageThresh1_Inp()</code> function compares each pixel in the image to a threshold value on a per-channel basis. If the pixel is less than or equal to the threshold value, then it is reset to the low output level. If the pixel is greater than the threshold value, then it is reset to the high output level.</p> <p>It uses the following equation:</p> <pre>srcdst[x][y][i] = glow[i] if srcdst[x][y][i] ≤ thresh[i] srcdst[x][y][i] = ghigh[i] if srcdst[x][y][i] > thresh[i]</pre>								
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>srcdst</i></td><td>Pointer to source and destination image.</td></tr><tr><td><i>thresh</i></td><td>Threshold value. <code>thresh[i]</code> contains the threshold for channel <i>i</i>.</td></tr><tr><td><i>ghigh</i></td><td>High output level. <code>ghigh[i]</code> contains the high output level for channel <i>i</i>.</td></tr><tr><td><i>glow</i></td><td>Low output level. <code>glow[i]</code> contains the low output level for channel <i>i</i>.</td></tr></table>	<i>srcdst</i>	Pointer to source and destination image.	<i>thresh</i>	Threshold value. <code>thresh[i]</code> contains the threshold for channel <i>i</i> .	<i>ghigh</i>	High output level. <code>ghigh[i]</code> contains the high output level for channel <i>i</i> .	<i>glow</i>	Low output level. <code>glow[i]</code> contains the low output level for channel <i>i</i> .
<i>srcdst</i>	Pointer to source and destination image.								
<i>thresh</i>	Threshold value. <code>thresh[i]</code> contains the threshold for channel <i>i</i> .								
<i>ghigh</i>	High output level. <code>ghigh[i]</code> contains the high output level for channel <i>i</i> .								
<i>glow</i>	Low output level. <code>glow[i]</code> contains the low output level for channel <i>i</i> .								
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .								
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:								
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe		
ATTRIBUTE TYPE	ATTRIBUTE VALUE								
Interface Stability	Evolving								
MT-Level	MT-Safe								
SEE ALSO	<code>mllib_ImageThresh1(3MLIB)</code> , <code>mllib_ImageThresh1_Fp(3MLIB)</code> , <code>mllib_ImageThresh1_Fp_Inp(3MLIB)</code> , <code>mllib_ImageThresh2(3MLIB)</code> , <code>mllib_ImageThresh2_Fp(3MLIB)</code> , <code>mllib_ImageThresh2_Fp_Inp(3MLIB)</code> , <code>mllib_ImageThresh2_Inp(3MLIB)</code> , <code>mllib_ImageThresh3(3MLIB)</code> , <code>mllib_ImageThresh3_Fp(3MLIB)</code> , <code>mllib_ImageThresh3_Fp_Inp(3MLIB)</code> , <code>mllib_ImageThresh3_Inp(3MLIB)</code> , <code>mllib_ImageThresh4(3MLIB)</code> , <code>mllib_ImageThresh4_Fp(3MLIB)</code> , <code>mllib_ImageThresh4_Fp_Inp(3MLIB)</code> , <code>mllib_ImageThresh4_Inp(3MLIB)</code> , <code>mllib_ImageThresh5(3MLIB)</code> , <code>mllib_ImageThresh5_Fp(3MLIB)</code> , <code>mllib_ImageThresh5_Fp_Inp(3MLIB)</code> , <code>mllib_ImageThresh5_Inp(3MLIB)</code> , <code>attributes(5)</code>								

NAME mllib_ImageThresh2 – image thresholding

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageThresh2(mllib_image *dst, const mllib_image
    *src, const mllib_s32 *thresh, const mllib_s32 *glow);
```

DESCRIPTION The mllib_ImageThresh2() function compares each pixel in the source image to a threshold value. If the pixel is less than or equal to the threshold value, then the destination pixel is set to the low output level. If the pixel is greater than the threshold value, then the destination pixel is set to the value of the source pixel.

It uses the following equation:

```
dst[x][y][i] = glow[i]          if src[x][y][i] ≤ thresh[i]
dst[x][y][i] = src[x][y][i]    if src[x][y][i] > thresh[i]
```

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

thresh Threshold value. thresh[i] contains the threshold for channel i.

glow Low output level. glow[i] contains the low output level for channel i.

RETURN VALUES The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageThresh1(3MLIB), mllib_ImageThresh1_Fp(3MLIB), mllib_ImageThresh1_Fp_Inp(3MLIB), mllib_ImageThresh1_Inp(3MLIB), mllib_ImageThresh2_Fp(3MLIB), mllib_ImageThresh2_Fp_Inp(3MLIB), mllib_ImageThresh2_Inp(3MLIB), mllib_ImageThresh3(3MLIB), mllib_ImageThresh3_Fp(3MLIB), mllib_ImageThresh3_Fp_Inp(3MLIB), mllib_ImageThresh3_Inp(3MLIB), mllib_ImageThresh4(3MLIB), mllib_ImageThresh4_Fp(3MLIB), mllib_ImageThresh4_Fp_Inp(3MLIB), mllib_ImageThresh4_Inp(3MLIB), mllib_ImageThresh5(3MLIB), mllib_ImageThresh5_Fp(3MLIB), mllib_ImageThresh5_Fp_Inp(3MLIB), mllib_ImageThresh5_Inp(3MLIB), attributes(5)

mllib_ImageThresh2_Fp(3MLIB)

NAME	mllib_ImageThresh2_Fp – image thresholding						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageThresh2_Fp(mllib_image *dst, const mllib_image *src, const mllib_d64 *thresh, const mllib_d64 *glow);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageThresh2_Fp()</code> function compares each pixel in the floating-point source image to a threshold value. If the pixel is less than or equal to the threshold value, then the destination pixel is set to the low output level. If the pixel is greater than the threshold value, then the destination pixel is set to the value of the source pixel.</p> <p>It uses the following equation:</p> $\begin{aligned} \text{dst}[x][y][i] &= \text{glow}[i] && \text{if } \text{src}[x][y][i] \leq \text{thresh}[i] \\ \text{dst}[x][y][i] &= \text{src}[x][y][i] && \text{if } \text{src}[x][y][i] > \text{thresh}[i] \end{aligned}$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>thresh</i> Threshold value. <code>thresh[i]</code> contains the threshold for channel <i>i</i>.</p> <p><i>glow</i> Low output level. <code>glow[i]</code> contains the low output level for channel <i>i</i>.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageThresh1(3MLIB)</code> , <code>mllib_ImageThresh1_Fp(3MLIB)</code> , <code>mllib_ImageThresh1_Fp_Inp(3MLIB)</code> , <code>mllib_ImageThresh1_Inp(3MLIB)</code> , <code>mllib_ImageThresh2(3MLIB)</code> , <code>mllib_ImageThresh2_Fp_Inp(3MLIB)</code> , <code>mllib_ImageThresh2_Inp(3MLIB)</code> , <code>mllib_ImageThresh3(3MLIB)</code> , <code>mllib_ImageThresh3_Fp(3MLIB)</code> , <code>mllib_ImageThresh3_Fp_Inp(3MLIB)</code> , <code>mllib_ImageThresh3_Inp(3MLIB)</code> , <code>mllib_ImageThresh4(3MLIB)</code> , <code>mllib_ImageThresh4_Fp(3MLIB)</code> , <code>mllib_ImageThresh4_Fp_Inp(3MLIB)</code> , <code>mllib_ImageThresh4_Inp(3MLIB)</code> , <code>mllib_ImageThresh5(3MLIB)</code> , <code>mllib_ImageThresh5_Fp(3MLIB)</code> , <code>mllib_ImageThresh5_Fp_Inp(3MLIB)</code> , <code>mllib_ImageThresh5_Inp(3MLIB)</code> , <code>attributes(5)</code>						

NAME | mllib_ImageThresh2_Fp_Inp – image thresholding

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageThresh2_Fp_Inp(mllib_image *srcdst, const
mllib_d64 *thresh, const mllib_d64 *glow);
```

DESCRIPTION | The mllib_ImageThresh2_Fp_Inp() function compares each pixel in the floating-point source image to a threshold value, in place. If the pixel is less than or equal to the threshold value, then the destination pixel is set to the low output level. If the pixel is greater than the threshold value, then the destination pixel is set to the value of the source pixel.

It uses the following equation:

$$srcdst[x][y][i] = glow[i] \quad \text{if } srcdst[x][y][i] \leq thresh[i]$$

PARAMETERS | The function takes the following arguments:

srcdst | Pointer to source and destination image.

thresh | Threshold value. *thresh[i]* contains the threshold for channel *i*.

glow | Low output level. *glow[i]* contains the low output level for channel *i*.

RETURN VALUES | The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_ImageThresh1(3MLIB), mllib_ImageThresh1_Fp(3MLIB), mllib_ImageThresh1_Fp_Inp(3MLIB), mllib_ImageThresh1_Inp(3MLIB), mllib_ImageThresh2(3MLIB), mllib_ImageThresh2_Fp(3MLIB), mllib_ImageThresh2_Inp(3MLIB), mllib_ImageThresh3(3MLIB), mllib_ImageThresh3_Fp(3MLIB), mllib_ImageThresh3_Fp_Inp(3MLIB), mllib_ImageThresh3_Inp(3MLIB), mllib_ImageThresh4(3MLIB), mllib_ImageThresh4_Fp(3MLIB), mllib_ImageThresh4_Fp_Inp(3MLIB), mllib_ImageThresh4_Inp(3MLIB), mllib_ImageThresh5(3MLIB), mllib_ImageThresh5_Fp(3MLIB), mllib_ImageThresh5_Fp_Inp(3MLIB), mllib_ImageThresh5_Inp(3MLIB), attributes(5)

mllib_ImageThresh2_Inp(3MLIB)

NAME	mllib_ImageThresh2_Inp – image thresholding						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageThresh2_Inp(mllib_image *<i>srcdst</i>, const mllib_s32 *<i>thresh</i>, const mllib_s32 *<i>glow</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageThresh2_Inp()</code> function compares each pixel in the source image to a threshold value, in place. If the pixel is less than or equal to the threshold value, then the destination pixel is set to the low output level. If the pixel is greater than the threshold value, then the destination pixel is set to the value of the source pixel.</p> <p>It uses the following equation:</p> $\text{srcdst}[x][y][i] = \text{glow}[i] \quad \text{if } \text{srcdst}[x][y][i] \leq \text{thresh}[i]$						
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>srcdst</i></td><td>Pointer to source and destination image.</td></tr><tr><td><i>thresh</i></td><td>Threshold value. <code>thresh[i]</code> contains the threshold for channel <i>i</i>.</td></tr><tr><td><i>glow</i></td><td>Low output level. <code>glow[i]</code> contains the low output level for channel <i>i</i>.</td></tr></table>	<i>srcdst</i>	Pointer to source and destination image.	<i>thresh</i>	Threshold value. <code>thresh[i]</code> contains the threshold for channel <i>i</i> .	<i>glow</i>	Low output level. <code>glow[i]</code> contains the low output level for channel <i>i</i> .
<i>srcdst</i>	Pointer to source and destination image.						
<i>thresh</i>	Threshold value. <code>thresh[i]</code> contains the threshold for channel <i>i</i> .						
<i>glow</i>	Low output level. <code>glow[i]</code> contains the low output level for channel <i>i</i> .						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageThresh1(3MLIB)</code> , <code>mllib_ImageThresh1_Fp(3MLIB)</code> , <code>mllib_ImageThresh1_Fp_Inp(3MLIB)</code> , <code>mllib_ImageThresh1_Inp(3MLIB)</code> , <code>mllib_ImageThresh2(3MLIB)</code> , <code>mllib_ImageThresh2_Fp(3MLIB)</code> , <code>mllib_ImageThresh2_Fp_Inp(3MLIB)</code> , <code>mllib_ImageThresh3(3MLIB)</code> , <code>mllib_ImageThresh3_Fp(3MLIB)</code> , <code>mllib_ImageThresh3_Fp_Inp(3MLIB)</code> , <code>mllib_ImageThresh3_Inp(3MLIB)</code> , <code>mllib_ImageThresh4(3MLIB)</code> , <code>mllib_ImageThresh4_Fp(3MLIB)</code> , <code>mllib_ImageThresh4_Fp_Inp(3MLIB)</code> , <code>mllib_ImageThresh4_Inp(3MLIB)</code> , <code>mllib_ImageThresh5(3MLIB)</code> , <code>mllib_ImageThresh5_Fp(3MLIB)</code> , <code>mllib_ImageThresh5_Fp_Inp(3MLIB)</code> , <code>mllib_ImageThresh5_Inp(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageThresh3 – image thresholding

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageThresh3(mllib_image *dst, const mllib_image
    *src, const mllib_s32 *thresh, const mllib_s32 *ghigh);
```

DESCRIPTION The `mllib_ImageThresh3()` function compares each pixel in the source image to a threshold value. If the pixel is less than or equal to the threshold value, then the destination pixel is set to the value of the source pixel. If the pixel is greater than the threshold value, then the destination pixel is set to the high output level.

It uses the following equation:

```
dst[x][y][i] = src[x][y][i]  if src[x][y][i] ≤ thresh[i]
dst[x][y][i] = ghigh[i]     if src[x][y][i] > thresh[i]
```

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

thresh Threshold value. `thresh[i]` contains the threshold for channel *i*.

ghigh High output level. `ghigh[i]` contains the high output level for channel *i*.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageThresh1(3MLIB)`, `mllib_ImageThresh1_Fp(3MLIB)`, `mllib_ImageThresh1_Fp_Inp(3MLIB)`, `mllib_ImageThresh1_Inp(3MLIB)`, `mllib_ImageThresh2(3MLIB)`, `mllib_ImageThresh2_Fp(3MLIB)`, `mllib_ImageThresh2_Fp_Inp(3MLIB)`, `mllib_ImageThresh2_Inp(3MLIB)`, `mllib_ImageThresh3_Fp(3MLIB)`, `mllib_ImageThresh3_Fp_Inp(3MLIB)`, `mllib_ImageThresh3_Inp(3MLIB)`, `mllib_ImageThresh4(3MLIB)`, `mllib_ImageThresh4_Fp(3MLIB)`, `mllib_ImageThresh4_Fp_Inp(3MLIB)`, `mllib_ImageThresh4_Inp(3MLIB)`, `mllib_ImageThresh5(3MLIB)`, `mllib_ImageThresh5_Fp(3MLIB)`, `mllib_ImageThresh5_Fp_Inp(3MLIB)`, `mllib_ImageThresh5_Inp(3MLIB)`, `attributes(5)`

mllib_ImageThresh3_Fp(3MLIB)

NAME	mllib_ImageThresh3_Fp – image thresholding						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageThresh3_Fp(mllib_image *dst, const mllib_image *src, const mllib_d64 *thresh, const mllib_d64 *ghigh);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageThresh3_Fp()</code> function compares each pixel in the floating-point source image to a threshold value. If the pixel is less than or equal to the threshold value, then the destination pixel is set to the value of the source pixel. If the pixel is greater than the threshold value, then the destination pixel is set to the high output level.</p> <p>It uses the following equation:</p> <pre>dst[x][y][i] = src[x][y][i] if src[x][y][i] ≤ thresh[i] dst[x][y][i] = ghigh[i] if src[x][y][i] > thresh[i]</pre>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>thresh</i> Threshold value. <code>thresh[i]</code> contains the threshold for channel <i>i</i>.</p> <p><i>ghigh</i> High output level. <code>ghigh[i]</code> contains the high output level for channel <i>i</i>.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<p><code>mllib_ImageThresh1(3MLIB)</code>, <code>mllib_ImageThresh1_Fp(3MLIB)</code>, <code>mllib_ImageThresh1_Fp_Inp(3MLIB)</code>, <code>mllib_ImageThresh1_Inp(3MLIB)</code>, <code>mllib_ImageThresh2(3MLIB)</code>, <code>mllib_ImageThresh2_Fp(3MLIB)</code>, <code>mllib_ImageThresh2_Fp_Inp(3MLIB)</code>, <code>mllib_ImageThresh2_Inp(3MLIB)</code>, <code>mllib_ImageThresh3(3MLIB)</code>, <code>mllib_ImageThresh3_Fp_Inp(3MLIB)</code>, <code>mllib_ImageThresh3_Inp(3MLIB)</code>, <code>mllib_ImageThresh4(3MLIB)</code>, <code>mllib_ImageThresh4_Fp(3MLIB)</code>, <code>mllib_ImageThresh4_Fp_Inp(3MLIB)</code>, <code>mllib_ImageThresh4_Inp(3MLIB)</code>, <code>mllib_ImageThresh5(3MLIB)</code>, <code>mllib_ImageThresh5_Fp(3MLIB)</code>, <code>mllib_ImageThresh5_Fp_Inp(3MLIB)</code>, <code>mllib_ImageThresh5_Inp(3MLIB)</code>, <code>attributes(5)</code></p>						

NAME mllib_ImageThresh3_Fp_Inp – image thresholding

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageThresh3_Fp_Inp(mllib_image *srcdst, const
    mllib_d64 *thresh, const mllib_d64 *ghigh);
```

DESCRIPTION The `mllib_ImageThresh3_Fp_Inp()` function compares each pixel in the floating-point source image to a threshold value, in place. If the pixel is less than or equal to the threshold value, then the destination pixel is set to the value of the source pixel. If the pixel is greater than the threshold value, then the destination pixel is set to the high output level.

It uses the following equation:

$$srcdst[x][y][i] = ghigh[i] \text{ if } srcdst[x][y][i] > thresh[i]$$

PARAMETERS The function takes the following arguments:

srcdst Pointer to source and destination image.

thresh Threshold value. `thresh[i]` contains the threshold for channel *i*.

ghigh High output level. `ghigh[i]` contains the high output level for channel *i*.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageThresh1(3MLIB)`, `mllib_ImageThresh1_Fp(3MLIB)`, `mllib_ImageThresh1_Fp_Inp(3MLIB)`, `mllib_ImageThresh1_Inp(3MLIB)`, `mllib_ImageThresh2(3MLIB)`, `mllib_ImageThresh2_Fp(3MLIB)`, `mllib_ImageThresh2_Fp_Inp(3MLIB)`, `mllib_ImageThresh2_Inp(3MLIB)`, `mllib_ImageThresh3(3MLIB)`, `mllib_ImageThresh3_Fp(3MLIB)`, `mllib_ImageThresh3_Inp(3MLIB)`, `mllib_ImageThresh4(3MLIB)`, `mllib_ImageThresh4_Fp(3MLIB)`, `mllib_ImageThresh4_Fp_Inp(3MLIB)`, `mllib_ImageThresh4_Inp(3MLIB)`, `mllib_ImageThresh5(3MLIB)`, `mllib_ImageThresh5_Fp(3MLIB)`, `mllib_ImageThresh5_Fp_Inp(3MLIB)`, `mllib_ImageThresh5_Inp(3MLIB)`, `attributes(5)`

mllib_ImageThresh3_Inp(3MLIB)

NAME	mllib_ImageThresh3_Inp – image thresholding						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageThresh3_Inp(mllib_image *<i>srcdst</i>, const mllib_s32 *<i>thresh</i>, const mllib_s32 *<i>ghigh</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageThresh3_Inp()</code> function compares each pixel in the source image to a threshold value, in place. If the pixel is less than or equal to the threshold value, then the destination pixel is set to the value of the source pixel. If the pixel is greater than the threshold value, then the destination pixel is set to the high output level.</p> <p>It uses the following equation:</p> $\text{srcdst}[x][y][i] = \text{ghigh}[i] \quad \text{if } \text{srcdst}[x][y][i] > \text{thresh}[i]$						
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>srcdst</i></td><td>Pointer to source and destination image.</td></tr><tr><td><i>thresh</i></td><td>Threshold value. <code>thresh[i]</code> contains the threshold for channel <i>i</i>.</td></tr><tr><td><i>ghigh</i></td><td>High output level. <code>ghigh[i]</code> contains the high output level for channel <i>i</i>.</td></tr></table>	<i>srcdst</i>	Pointer to source and destination image.	<i>thresh</i>	Threshold value. <code>thresh[i]</code> contains the threshold for channel <i>i</i> .	<i>ghigh</i>	High output level. <code>ghigh[i]</code> contains the high output level for channel <i>i</i> .
<i>srcdst</i>	Pointer to source and destination image.						
<i>thresh</i>	Threshold value. <code>thresh[i]</code> contains the threshold for channel <i>i</i> .						
<i>ghigh</i>	High output level. <code>ghigh[i]</code> contains the high output level for channel <i>i</i> .						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageThresh1(3MLIB)</code> , <code>mllib_ImageThresh1_Fp(3MLIB)</code> , <code>mllib_ImageThresh1_Fp_Inp(3MLIB)</code> , <code>mllib_ImageThresh1_Inp(3MLIB)</code> , <code>mllib_ImageThresh2(3MLIB)</code> , <code>mllib_ImageThresh2_Fp(3MLIB)</code> , <code>mllib_ImageThresh2_Fp_Inp(3MLIB)</code> , <code>mllib_ImageThresh2_Inp(3MLIB)</code> , <code>mllib_ImageThresh3(3MLIB)</code> , <code>mllib_ImageThresh3_Fp(3MLIB)</code> , <code>mllib_ImageThresh3_Fp_Inp(3MLIB)</code> , <code>mllib_ImageThresh4(3MLIB)</code> , <code>mllib_ImageThresh4_Fp(3MLIB)</code> , <code>mllib_ImageThresh4_Fp_Inp(3MLIB)</code> , <code>mllib_ImageThresh4_Inp(3MLIB)</code> , <code>mllib_ImageThresh5(3MLIB)</code> , <code>mllib_ImageThresh5_Fp(3MLIB)</code> , <code>mllib_ImageThresh5_Fp_Inp(3MLIB)</code> , <code>mllib_ImageThresh5_Inp(3MLIB)</code> , <code>attributes(5)</code>						

NAME	mlib_ImageThresh4 – image thresholding						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmlib [<i>library...</i>] #include <mlib.h> mlib_status mlib_ImageThresh4(mlib_image *<i>dst</i>, const mlib_image *<i>src</i>, const mlib_s32 *<i>thigh</i>, const mlib_s32 *<i>tlow</i>, const mlib_s32 *<i>ghigh</i>, const mlib_s32 *<i>glow</i>);</pre>						
DESCRIPTION	<p>The <code>mlib_ImageThresh4()</code> function compares each pixel in the source image to two threshold values, <i>tlow</i> and <i>thigh</i>. If the pixel is less than the lower threshold value, <i>tlow</i>, then the destination pixel is set to the lower output level, <i>glow</i>. If the pixel is greater than the higher threshold value, <i>thigh</i>, then the destination pixel is set to the higher output level, <i>ghigh</i>. Otherwise, the destination pixel is set to the value of the source pixel.</p> <p>It uses the following equation:</p> <pre>dst[x][y][i] = glow[i] if src[x][y][i] < tlow[i] dst[x][y][i] = src[x][y][i] if tlow[i] ≤ src[x][y][i] ≤ thigh[i] dst[x][y][i] = ghigh[i] if src[x][y][i] > thigh[i]</pre>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>thigh</i> High threshold value. <code>thigh[i]</code> holds the high threshold for channel <i>i</i>.</p> <p><i>tlow</i> Low threshold value. <code>tlow[i]</code> holds the low threshold for channel <i>i</i>.</p> <p><i>ghigh</i> High output grayscale level. <code>ghigh[i]</code> holds the high output grayscale level for channel <i>i</i>.</p> <p><i>glow</i> Low output grayscale level. <code>glow[i]</code> holds the low output grayscale level for channel <i>i</i>.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mlib_ImageThresh1(3MLIB)</code> , <code>mlib_ImageThresh1_Fp(3MLIB)</code> , <code>mlib_ImageThresh1_Fp_Inp(3MLIB)</code> , <code>mlib_ImageThresh1_Inp(3MLIB)</code> , <code>mlib_ImageThresh2(3MLIB)</code> , <code>mlib_ImageThresh2_Fp(3MLIB)</code> ,						

`mllib_ImageThresh4(3MLIB)`

```
mllib_ImageThresh2_Fp_Inp(3MLIB), mllib_ImageThresh2_Inp(3MLIB),  
mllib_ImageThresh3(3MLIB), mllib_ImageThresh3_Fp(3MLIB),  
mllib_ImageThresh3_Fp_Inp(3MLIB), mllib_ImageThresh3_Inp(3MLIB),  
mllib_ImageThresh4_Fp(3MLIB), mllib_ImageThresh4_Fp_Inp(3MLIB),  
mllib_ImageThresh4_Inp(3MLIB), mllib_ImageThresh5(3MLIB),  
mllib_ImageThresh5_Fp(3MLIB), mllib_ImageThresh5_Fp_Inp(3MLIB),  
mllib_ImageThresh5_Inp(3MLIB), attributes(5)
```

NAME mllib_ImageThresh4_Fp – image thresholding

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageThresh4_Fp(mllib_image *dst, const
    mllib_image *src, const mllib_d64 *thigh, const mllib_d64 *tlow,
    const mllib_d64 *ghigh, const mllib_d64 *glow);
```

DESCRIPTION The mllib_ImageThresh4_Fp() function compares each pixel in the source image to two threshold values, *tlow* and *thigh*. If the pixel is less than the lower threshold value, *tlow*, then the destination pixel is set to the lower output level, *glow*. If the pixel is greater than the higher threshold value, *thigh*, then the destination pixel is set to the higher output level, *ghigh*. Otherwise, the destination pixel is set to the value of the source pixel.

It uses the following equation:

```
dst[x][y][i] = glow[i]          if src[x][y][i] < tlow[i]
dst[x][y][i] = src[x][y][i]    if tlow[i] ≤ src[x][y][i] ≤ thigh[i]
dst[x][y][i] = ghigh[i]       if src[x][y][i] > thigh[i]
```

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

thigh High threshold value. *thigh[i]* holds the high threshold for channel *i*.

tlow Low threshold value. *tlow[i]* holds the low threshold for channel *i*.

ghigh High output grayscale level. *ghigh[i]* holds the high output grayscale level for channel *i*.

glow Low output grayscale level. *glow[i]* holds the low output grayscale level for channel *i*.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageThresh1(3MLIB), mllib_ImageThresh1_Fp(3MLIB), mllib_ImageThresh1_Fp_Inp(3MLIB), mllib_ImageThresh1_Inp(3MLIB), mllib_ImageThresh2(3MLIB), mllib_ImageThresh2_Fp(3MLIB),

`mlib_ImageThresh4_Fp(3MLIB)`

```
mlib_ImageThresh2_Fp_Inp(3MLIB), mlib_ImageThresh2_Inp(3MLIB),  
mlib_ImageThresh3(3MLIB), mlib_ImageThresh3_Fp(3MLIB),  
mlib_ImageThresh3_Fp_Inp(3MLIB), mlib_ImageThresh3_Inp(3MLIB),  
mlib_ImageThresh4(3MLIB), mlib_ImageThresh4_Fp_Inp(3MLIB),  
mlib_ImageThresh4_Inp(3MLIB), mlib_ImageThresh5(3MLIB),  
mlib_ImageThresh5_Fp(3MLIB), mlib_ImageThresh5_Fp_Inp(3MLIB),  
mlib_ImageThresh5_Inp(3MLIB), attributes(5)
```

NAME mllib_ImageThresh4_Fp_Inp – image thresholding

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageThresh4_Fp_Inp(mllib_image *srcdst, const
    mllib_d64 *thigh, const mllib_d64 *tlow, const mllib_d64 *ghigh,
    const mllib_d64 *glow);
```

DESCRIPTION The mllib_ImageThresh4_Fp_Inp() function compares each pixel in the source image to two threshold values, *tlow* and *thigh*. If the pixel is less than the lower threshold value, *tlow*, then the destination pixel is set to the lower output level, *glow*. If the pixel is greater than the higher threshold value, *thigh*, then the destination pixel is set to the higher output level, *ghigh*. Otherwise, the destination pixel is set to the value of the source pixel.

It uses the following equation:

```
srcdst[x][y][i] = glow[i]    if srcdst[x][y][i] < tlow[i]
srcdst[x][y][i] = ghigh[i]   if srcdst[x][y][i] > thigh[i]
```

PARAMETERS The function takes the following arguments:

srcdst Pointer to source and destination image.

thigh High threshold value. *thigh[i]* holds the high threshold for channel *i*.

tlow Low threshold value. *tlow[i]* holds the low threshold for channel *i*.

ghigh High output grayscale level. *ghigh[i]* holds the high output grayscale level for channel *i*.

glow Low output grayscale level. *glow[i]* holds the low output grayscale level for channel *i*.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageThresh1(3MLIB), mllib_ImageThresh1_Fp(3MLIB), mllib_ImageThresh1_Fp_Inp(3MLIB), mllib_ImageThresh1_Inp(3MLIB), mllib_ImageThresh2(3MLIB), mllib_ImageThresh2_Fp(3MLIB), mllib_ImageThresh2_Fp_Inp(3MLIB), mllib_ImageThresh2_Inp(3MLIB), mllib_ImageThresh3(3MLIB), mllib_ImageThresh3_Fp(3MLIB), mllib_ImageThresh3_Fp_Inp(3MLIB), mllib_ImageThresh3_Inp(3MLIB),

`mlib_ImageThresh4_Fp_Inp(3MLIB)`

```
mlib_ImageThresh4(3MLIB), mlib_ImageThresh4_Fp(3MLIB),  
mlib_ImageThresh4_Inp(3MLIB), mlib_ImageThresh5(3MLIB),  
mlib_ImageThresh5_Fp(3MLIB), mlib_ImageThresh5_Fp_Inp(3MLIB),  
mlib_ImageThresh5_Inp(3MLIB), attributes(5)
```

NAME mllib_ImageThresh4_Inp – image thresholding

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageThresh4_Inp(mllib_image *srcdst, const
    mllib_s32 *thigh, const mllib_s32 *tlow, const mllib_s32 *ghigh,
    const mllib_s32 *glow);
```

DESCRIPTION The mllib_ImageThresh4_Inp() function compares each pixel in the source image to two threshold values, *tlow* and *thigh*. If the pixel is less than the lower threshold value, *tlow*, then the destination pixel is set to the lower output level, *glow*. If the pixel is greater than the higher threshold value, *thigh*, then the destination pixel is set to the higher output level, *ghigh*. Otherwise, the destination pixel is set to the value of the source pixel.

It uses the following equation:

```
srcdst[x][y][i] = glow[i]    if srcdst[x][y][i] < tlow[i]
srcdst[x][y][i] = ghigh[i]   if srcdst[x][y][i] > thigh[i]
```

PARAMETERS The function takes the following arguments:

srcdst Pointer to source and destination image.

thigh High threshold value. *thigh[i]* holds the high threshold for channel *i*.

tlow Low threshold value. *tlow[i]* holds the low threshold for channel *i*.

ghigh High output grayscale level. *ghigh[i]* holds the high output grayscale level for channel *i*.

glow Low output grayscale level. *glow[i]* holds the low output grayscale level for channel *i*.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageThresh1(3MLIB), mllib_ImageThresh1_Fp(3MLIB), mllib_ImageThresh1_Fp_Inp(3MLIB), mllib_ImageThresh1_Inp(3MLIB), mllib_ImageThresh2(3MLIB), mllib_ImageThresh2_Fp(3MLIB), mllib_ImageThresh2_Fp_Inp(3MLIB), mllib_ImageThresh2_Inp(3MLIB), mllib_ImageThresh3(3MLIB), mllib_ImageThresh3_Fp(3MLIB), mllib_ImageThresh3_Fp_Inp(3MLIB), mllib_ImageThresh3_Inp(3MLIB),

`mlib_ImageThresh4_Inp(3MLIB)`

```
mlib_ImageThresh4(3MLIB), mlib_ImageThresh4_Fp(3MLIB),  
mlib_ImageThresh4_Fp_Inp(3MLIB), mlib_ImageThresh5(3MLIB),  
mlib_ImageThresh5_Fp(3MLIB), mlib_ImageThresh5_Fp_Inp(3MLIB),  
mlib_ImageThresh5_Inp(3MLIB), attributes(5)
```


NAME	mllib_ImageThresh5 – image thresholding						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageThresh5(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, const mllib_s32 *<i>thigh</i>, const mllib_s32 *<i>tlow</i>, const mllib_s32 *<i>gray</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageThresh5()</code> function compares each pixel in the source image to two threshold values, <i>tlow</i> and <i>thigh</i>. If the pixel is in between the lower threshold value, <i>tlow</i>, and the higher threshold value, <i>thigh</i>, (inclusive on both sides), then the destination pixel is set to the value <i>gray</i>. Otherwise, the destination pixel is set to the value of the source pixel.</p> <p>It uses the following equation:</p> <pre>dst[x][y][i] = src[x][y][i] if src[x][y][i] < tlow[i] dst[x][y][i] = gray[i] if tlow[i] ≤ src[x][y][i] ≤ thigh[i] dst[x][y][i] = src[x][y][i] if src[x][y][i] > thigh[i]</pre>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>thigh</i> High threshold value. <code>thigh[i]</code> holds the high threshold for channel <i>i</i>.</p> <p><i>tlow</i> Low threshold value. <code>tlow[i]</code> holds the low threshold for channel <i>i</i>.</p> <p><i>gray</i> Output grayscale level. <code>gray[i]</code> holds the output grayscale level for channel <i>i</i>.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<p><code>mllib_ImageThresh1(3MLIB)</code>, <code>mllib_ImageThresh1_Fp(3MLIB)</code>, <code>mllib_ImageThresh1_Fp_Inp(3MLIB)</code>, <code>mllib_ImageThresh1_Inp(3MLIB)</code>, <code>mllib_ImageThresh2(3MLIB)</code>, <code>mllib_ImageThresh2_Fp(3MLIB)</code>, <code>mllib_ImageThresh2_Fp_Inp(3MLIB)</code>, <code>mllib_ImageThresh2_Inp(3MLIB)</code>, <code>mllib_ImageThresh3(3MLIB)</code>, <code>mllib_ImageThresh3_Fp(3MLIB)</code>, <code>mllib_ImageThresh3_Fp_Inp(3MLIB)</code>, <code>mllib_ImageThresh3_Inp(3MLIB)</code>, <code>mllib_ImageThresh4(3MLIB)</code>, <code>mllib_ImageThresh4_Fp(3MLIB)</code>,</p>						

`mlib_ImageThresh5(3MLIB)`

```
mllib_ImageThresh4_Fp_Inp(3MLIB), mllib_ImageThresh4_Inp(3MLIB),  
mllib_ImageThresh5_Fp(3MLIB), mllib_ImageThresh5_Fp_Inp(3MLIB),  
mllib_ImageThresh5_Inp(3MLIB), attributes(5)
```

NAME | mllib_ImageThresh5_Fp – image thresholding

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageThresh5_Fp(mllib_image *dst, const
mllib_image *src, const mllib_d64 *thigh, const mllib_d64 *tlow,
const mllib_d64 *gray);
```

DESCRIPTION | The mllib_ImageThresh5_Fp() function compares each pixel in the source image to two threshold values, *tlow* and *thigh*. If the pixel is in between the lower threshold value, *tlow*, and the higher threshold value, *thigh*, (inclusive on both sides), then the destination pixel is set to the value *gray*. Otherwise, the destination pixel is set to the value of the source pixel.

It uses the following equation:

```
dst[x][y][i] = src[x][y][i] if src[x][y][i] < tlow[i]
dst[x][y][i] = gray[i] if tlow[i] ≤ src[x][y][i] ≤ thigh[i]
dst[x][y][i] = src[x][y][i] if src[x][y][i] > thigh[i]
```

PARAMETERS | The function takes the following arguments:

dst | Pointer to destination image.

src | Pointer to source image.

thigh | High threshold value. thigh[i] holds the high threshold for channel i.

tlow | Low threshold value. tlow[i] holds the low threshold for channel i.

gray | Output grayscale level. gray[i] holds the output grayscale level for channel i.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_ImageThresh1(3MLIB), mllib_ImageThresh1_Fp(3MLIB), mllib_ImageThresh1_Fp_Inp(3MLIB), mllib_ImageThresh1_Inp(3MLIB), mllib_ImageThresh2(3MLIB), mllib_ImageThresh2_Fp(3MLIB), mllib_ImageThresh2_Fp_Inp(3MLIB), mllib_ImageThresh2_Inp(3MLIB), mllib_ImageThresh3(3MLIB), mllib_ImageThresh3_Fp(3MLIB), mllib_ImageThresh3_Fp_Inp(3MLIB), mllib_ImageThresh3_Inp(3MLIB), mllib_ImageThresh4(3MLIB), mllib_ImageThresh4_Fp(3MLIB),

`mlib_ImageThresh5_Fp(3MLIB)`

```
mllib_ImageThresh4_Fp_Inp(3MLIB), mllib_ImageThresh4_Inp(3MLIB),  
mllib_ImageThresh5(3MLIB), mllib_ImageThresh5_Fp_Inp(3MLIB),  
mllib_ImageThresh5_Inp(3MLIB), attributes(5)
```

NAME | mllib_ImageThresh5_Fp_Inp – image thresholding

SYNOPSIS |

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageThresh5_Fp_Inp(mllib_image *srcdst, const
    mllib_d64 *thigh, const mllib_d64 *tlow, const mllib_d64 *gray);
```

DESCRIPTION | The mllib_ImageThresh5_Fp_Inp() function compares each pixel in the source image to two threshold values, *tlow* and *thigh*. If the pixel is in between the lower threshold value, *tlow*, and the higher threshold value, *thigh*, (inclusive on both sides), then the destination pixel is set to the value *gray*. Otherwise, the destination pixel is set to the value of the source pixel.

It uses the following equation:

$$srcdst[x][y][i] = gray[i] \quad \text{if } tlow[i] \leq srcdst[x][y][i] \leq thigh[i]$$

PARAMETERS | The function takes the following arguments:

srcdst | Pointer to source and destination image.

thigh | High threshold value. *thigh[i]* holds the high threshold for channel *i*.

tlow | Low threshold value. *tlow[i]* holds the low threshold for channel *i*.

gray | Output grayscale level. *gray[i]* holds the output grayscale level for channel *i*.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_ImageThresh1(3MLIB), mllib_ImageThresh1_Fp(3MLIB), mllib_ImageThresh1_Fp_Inp(3MLIB), mllib_ImageThresh1_Inp(3MLIB), mllib_ImageThresh2(3MLIB), mllib_ImageThresh2_Fp(3MLIB), mllib_ImageThresh2_Fp_Inp(3MLIB), mllib_ImageThresh2_Inp(3MLIB), mllib_ImageThresh3(3MLIB), mllib_ImageThresh3_Fp(3MLIB), mllib_ImageThresh3_Fp_Inp(3MLIB), mllib_ImageThresh3_Inp(3MLIB), mllib_ImageThresh4(3MLIB), mllib_ImageThresh4_Fp(3MLIB), mllib_ImageThresh4_Fp_Inp(3MLIB), mllib_ImageThresh4_Inp(3MLIB), mllib_ImageThresh5(3MLIB), mllib_ImageThresh5_Fp(3MLIB), mllib_ImageThresh5_Inp(3MLIB), attributes(5)

mllib_ImageThresh5_Inp(3MLIB)

NAME	mllib_ImageThresh5_Inp – image thresholding								
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageThresh5_Inp(mllib_image *srcdst, const mllib_s32 *thigh, const mllib_s32 *tlow, const mllib_s32 *gray);</pre>								
DESCRIPTION	<p>The <code>mllib_ImageThresh5_Inp()</code> function compares each pixel in the source image to two threshold values, <i>tlow</i> and <i>thigh</i>. If the pixel is in between the lower threshold value, <i>tlow</i>, and the higher threshold value, <i>thigh</i>, (inclusive on both sides), then the destination pixel is set to the value <i>gray</i>. Otherwise, the destination pixel is set to the value of the source pixel.</p> <p>It uses the following equation:</p> $\text{srcdst}[x][y][i] = \text{gray}[i] \quad \text{if } \text{tlow}[i] \leq \text{srcdst}[x][y][i] \leq \text{thigh}[i]$								
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>srcdst</i></td><td>Pointer to source and destination image.</td></tr><tr><td><i>thigh</i></td><td>High threshold value. <code>thigh[i]</code> holds the high threshold for channel <i>i</i>.</td></tr><tr><td><i>tlow</i></td><td>Low threshold value. <code>tlow[i]</code> holds the low threshold for channel <i>i</i>.</td></tr><tr><td><i>gray</i></td><td>Output grayscale level. <code>gray[i]</code> holds the output grayscale level for channel <i>i</i>.</td></tr></table>	<i>srcdst</i>	Pointer to source and destination image.	<i>thigh</i>	High threshold value. <code>thigh[i]</code> holds the high threshold for channel <i>i</i> .	<i>tlow</i>	Low threshold value. <code>tlow[i]</code> holds the low threshold for channel <i>i</i> .	<i>gray</i>	Output grayscale level. <code>gray[i]</code> holds the output grayscale level for channel <i>i</i> .
<i>srcdst</i>	Pointer to source and destination image.								
<i>thigh</i>	High threshold value. <code>thigh[i]</code> holds the high threshold for channel <i>i</i> .								
<i>tlow</i>	Low threshold value. <code>tlow[i]</code> holds the low threshold for channel <i>i</i> .								
<i>gray</i>	Output grayscale level. <code>gray[i]</code> holds the output grayscale level for channel <i>i</i> .								
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .								
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:								
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe		
ATTRIBUTE TYPE	ATTRIBUTE VALUE								
Interface Stability	Evolving								
MT-Level	MT-Safe								
SEE ALSO	<code>mllib_ImageThresh1(3MLIB)</code> , <code>mllib_ImageThresh1_Fp(3MLIB)</code> , <code>mllib_ImageThresh1_Fp_Inp(3MLIB)</code> , <code>mllib_ImageThresh1_Inp(3MLIB)</code> , <code>mllib_ImageThresh2(3MLIB)</code> , <code>mllib_ImageThresh2_Fp(3MLIB)</code> , <code>mllib_ImageThresh2_Fp_Inp(3MLIB)</code> , <code>mllib_ImageThresh2_Inp(3MLIB)</code> , <code>mllib_ImageThresh3(3MLIB)</code> , <code>mllib_ImageThresh3_Fp(3MLIB)</code> , <code>mllib_ImageThresh3_Fp_Inp(3MLIB)</code> , <code>mllib_ImageThresh3_Inp(3MLIB)</code> , <code>mllib_ImageThresh4(3MLIB)</code> , <code>mllib_ImageThresh4_Fp(3MLIB)</code> , <code>mllib_ImageThresh4_Fp_Inp(3MLIB)</code> , <code>mllib_ImageThresh4_Inp(3MLIB)</code> , <code>mllib_ImageThresh5(3MLIB)</code> , <code>mllib_ImageThresh5_Fp(3MLIB)</code> , <code>mllib_ImageThresh5_Fp_Inp(3MLIB)</code> , <code>attributes(5)</code>								

NAME	mlib_ImageXor – Xor						
SYNOPSIS	<pre>cc [flag...] file... -lmlib [library...] #include <mlib.h> mlib_status mlib_ImageXor(mlib_image *dst, const mlib_image *src1, const mlib_image *src2);</pre>						
DESCRIPTION	<p>The <code>mlib_ImageXor()</code> function computes the exclusive Or of the first source image with the second source image on a pixel-by-pixel basis.</p> <p>It uses the following equation:</p> $dst[x][y][i] = src1[x][y][i] \wedge src2[x][y][i]$ <p>The data type of the images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src1</i> Pointer to first source image.</p> <p><i>src2</i> Pointer to second source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mlib_ImageXor_Inp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageXor_Inp(3MLIB)

NAME	mllib_ImageXor_Inp – Xor, in place						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageXor_Inp(mllib_image *src1dst, const mllib_image *src2) ;</pre>						
DESCRIPTION	<p>The <code>mllib_ImageXor_Inp()</code> function computes the exclusive Or of the first source image with the second source image on a pixel-by-pixel basis, in place.</p> <p>It uses the following equation:</p> $\text{src1dst}[x][y][i] = \text{src1dst}[x][y][i] \wedge \text{src2}[x][y][i]$ <p>The data type of the images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>src1dst</i> Pointer to first source and destination image.</p> <p><i>src2</i> Pointer to second source image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageXor(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_ImageXProj – image X projection

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageXProj(mllib_d64 *xproj, const mllib_image
    *img);
```

DESCRIPTION The mllib_ImageXProj() function computes the sum of the pixels in each column of the source image.

The image must be a single-channel image.

It uses the following equation:

$$xproj[x] = \sum_{y=0}^{h-1} img[x][y][0]$$

where $x = 0, 1, \dots, w - 1$.

PARAMETERS The function takes the following arguments:

xproj Pointer to X-projection vector, where length is equal to the number of columns in the source image (in other words, the image width).

img Pointer to an input image.

RETURN VALUES The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageXProj_Fp(3MLIB), mllib_ImageYProj(3MLIB), mllib_ImageYProj_Fp(3MLIB), attributes(5)

mllib_ImageXProj_Fp(3MLIB)

NAME	mllib_ImageXProj_Fp – image X projection						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageXProj_Fp(mllib_d64 *xproj, const mllib_image *img);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageXProj_Fp()</code> function computes the sum of the pixels in each column of the floating-point source image.</p> <p>The image must be a single-channel image.</p> <p>It uses the following equation:</p> $xproj[x] = \sum_{y=0}^{h-1} img[x][y][0]$ <p>where $x = 0, 1, \dots, w - 1$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>xproj</i> Pointer to X-projection vector, where length is equal to the number of columns in the source image (in other words, the image width).</p> <p><i>img</i> Pointer to an input image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageXProj(3MLIB)</code> , <code>mllib_ImageYProj(3MLIB)</code> , <code>mllib_ImageYProj_Fp(3MLIB)</code> , <code>attributes(5)</code>						

NAME	mllib_ImageYProj – image Y projection						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageYProj(mllib_d64 *yproj, const mllib_image *img);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageYProj()</code> function computes the sum of the pixels in each row of the source image.</p> <p>The image must be a single-channel image.</p> <p>It uses the following equation:</p> $yproj[y] = \sum_{x=0}^{w-1} img[x][y][0]$ <p>where $y = 0, 1, \dots, h - 1$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>yproj</i> Pointer to Y-projection vector, where length is equal to the number of rows in the source image (in other words, the image height).</p> <p><i>img</i> Pointer to an input image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageXProj(3MLIB)</code> , <code>mllib_ImageXProj_Fp(3MLIB)</code> , <code>mllib_ImageYProj_Fp(3MLIB)</code> , <code>attributes(5)</code>						

mllib_ImageYProj_Fp(3MLIB)

NAME	mllib_ImageYProj_Fp – image Y projection						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageYProj_Fp(mllib_d64 *yproj, const mllib_image *img);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageYProj_Fp()</code> function computes the sum of the pixels in each row of the floating-point source image.</p> <p>The image must be a single-channel image.</p> <p>It uses the following equation:</p> $yproj[y] = \sum_{x=0}^{w-1} img[x][y][0]$ <p>where $y = 0, 1, \dots, h - 1$.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>yproj</i> Pointer to Y-projection vector, where length is equal to the number of rows in the source image (in other words, the image height).</p> <p><i>img</i> Pointer to an input image.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_ImageXProj(3MLIB)</code> , <code>mllib_ImageXProj_Fp(3MLIB)</code> , <code>mllib_ImageYProj(3MLIB)</code> , <code>attributes(5)</code>						

NAME	mlib_ImageZoom – zoom
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmlib [<i>library...</i>] #include <mlib.h> mlib_status mlib_ImageZoom(mlib_image *<i>dst</i>, const mlib_image *<i>src</i>, mlib_d64 <i>zoomx</i>, mlib_d64 <i>zoomy</i>, mlib_filter <i>filter</i>, mlib_edge <i>edge</i>);</pre>
DESCRIPTION	<p>The <code>mlib_ImageZoom()</code> function will enlarge or minify the source image by the X and Y zoom factors. It uses the interpolation method as described by the resampling filter.</p> <p>The data type of the images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p> <p>The center of the upper-left corner pixel of an image is located at (0.5, 0.5).</p> <p>The width and height of the destination image can be different from those of the source image.</p> <p>The center of the source image is mapped onto the center of the destination image.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>zoomx</i> X zoom factor. <code>zoomx > 0.0</code>.</p> <p><i>zoomy</i> Y zoom factor. <code>zoomy > 0.0</code>.</p> <p><i>filter</i> Type of resampling filter. It can be one of the following:</p> <p style="margin-left: 40px;"><code>MLIB_NEAREST</code> <code>MLIB_BILINEAR</code> <code>MLIB_BICUBIC</code> <code>MLIB_BICUBIC2</code></p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <p style="margin-left: 40px;"><code>MLIB_EDGE_DST_NO_WRITE</code> <code>MLIB_EDGE_DST_FILL_ZERO</code> <code>MLIB_EDGE_OP_NEAREST</code> <code>MLIB_EDGE_SRC_EXTEND</code> <code>MLIB_EDGE_SRC_EXTEND_INDEF</code> <code>MLIB_EDGE_SRC_PADDED</code></p>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .

mllib_ImageZoom(3MLIB)

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageZoom_Fp(3MLIB)`, `mllib_ImageZoomIn2X(3MLIB)`,
`mllib_ImageZoomIn2X_Fp(3MLIB)`, `mllib_ImageZoomIn2XIndex(3MLIB)`,
`mllib_ImageZoomIndex(3MLIB)`, `mllib_ImageZoomOut2X(3MLIB)`,
`mllib_ImageZoomOut2X_Fp(3MLIB)`, `mllib_ImageZoomOut2XIndex(3MLIB)`,
`attributes(5)`

NAME	mlib_ImageZoomBlend – image scaling with alpha blending
SYNOPSIS	<pre>cc [flag...] file... -lmlib [library...] #include <mlib.h> mlib_status mlib_ImageZoomBlend(mlib_image *dst, const mlib_image *src, mlib_d64 zoomx, mlib_d64 zoomy, mlib_filter filter, mlib_edge edge, mlib_blend blend, mlib_s32 alpha, mlib_s32 cmask);</pre>
DESCRIPTION	<p>The <code>mlib_ImageZoomBlend()</code> function will enlarge or minify the source image by the X and Y zoom factors and blend it with the destination image.</p> <p>This function is a special case of <code>mlib_ImageZoomTranslateBlend()</code> with the center of the source image being mapped to the center of the destination image.</p> <p>The center of the upper-left corner pixel of an image is considered to be located at (0.5, 0.5).</p> <p>Both <code>src</code> and <code>dst</code> must be of type <code>MLIB_BYTE</code>. They can have either 3 or 4 channels.</p> <p>The <code>src</code> image cannot have width or height larger than 32767.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to first source image.</p> <p><i>zoomx</i> X zoom factor. <code>zoomx > 0.0</code>.</p> <p><i>zoomy</i> Y zoom factor. <code>zoomy > 0.0</code>.</p> <p><i>filter</i> Type of resampling filter. It can be one of the following:</p> <p style="margin-left: 40px;"><code>MLIB_NEAREST</code> <code>MLIB_BILINEAR</code> <code>MLIB_BICUBIC</code> <code>MLIB_BICUBIC2</code></p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <p style="margin-left: 40px;"><code>MLIB_EDGE_DST_NO_WRITE</code> <code>MLIB_EDGE_DST_FILL_ZERO</code> <code>MLIB_EDGE_OP_NEAREST</code> <code>MLIB_EDGE_SRC_EXTEND</code> <code>MLIB_EDGE_SRC_EXTEND_INDEF</code> <code>MLIB_EDGE_SRC_PADDED</code></p> <p><i>blend</i> Type of alpha blending. It can be one of the following:</p> <p style="margin-left: 40px;"><code>MLIB_BLEND_GTK_SRC</code> <code>MLIB_BLEND_GTK_SRC_OVER</code> <code>MLIB_BLEND_GTK_SRC_OVER2</code></p> <p><i>alpha</i> Overall alpha for blending.</p> <p><i>cmask</i> Channel mask to indicate the alpha channel.</p>

`mlib_ImageZoomBlend(3MLIB)`

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mlib_ImageZoomTranslateBlend(3MLIB)`,
`mlib_ImageZoomTranslateTableBlend(3MLIB)`, `attributes(5)`

NAME mllib_ImageZoom_Fp – zoom

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageZoom_Fp(mllib_image *dst, const mllib_image
    *src, mllib_d64 zoomx, mllib_d64 zoomy, mllib_filter filter,
    mllib_edge edge);
```

DESCRIPTION The `mllib_ImageZoom_Fp()` function will enlarge or minify the floating-point source image by the X and Y zoom factors. It uses the interpolation method as described by the resampling filter.

The center of the upper-left corner pixel of an image is located at (0.5, 0.5).

The width and height of the destination image can be different from those of the source image.

The center of the source image is mapped onto the center of the destination image.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

zoomx X zoom factor. `zoomx > 0.0`.

zoomy Y zoom factor. `zoomy > 0.0`.

filter Type of resampling filter. It can be one of the following:

MLIB_NEAREST
MLIB_BILINEAR
MLIB_BICUBIC
MLIB_BICUBIC2

edge Type of edge condition. It can be one of the following:

MLIB_EDGE_DST_NO_WRITE
MLIB_EDGE_DST_FILL_ZERO
MLIB_EDGE_OP_NEAREST
MLIB_EDGE_SRC_EXTEND
MLIB_EDGE_SRC_EXTEND_INDEF
MLIB_EDGE_SRC_PADDED

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

mllib_ImageZoom_Fp(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	MT-Safe

SEE ALSO `mllib_ImageZoom(3MLIB)`, `mllib_ImageZoomIn2X(3MLIB)`,
`mllib_ImageZoomIn2X_Fp(3MLIB)`, `mllib_ImageZoomIn2XIndex(3MLIB)`,
`mllib_ImageZoomIndex(3MLIB)`, `mllib_ImageZoomOut2X(3MLIB)`,
`mllib_ImageZoomOut2X_Fp(3MLIB)`, `mllib_ImageZoomOut2XIndex(3MLIB)`,
`attributes(5)`

NAME mllib_ImageZoomIn2X – 2X zoom

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageZoomIn2X(mllib_image *dst, const mllib_image
*src, mllib_filter filter, mllib_edge edge);
```

DESCRIPTION The mllib_ImageZoomIn2X() function enlarges the source image by a factor of two. It uses the interpolation method as described by the resampling filter.

The center of the upper-left corner pixel of an image is located at (0.5, 0.5).

The width and height of the destination image can be different from those of the source image.

The center of the source image is mapped onto the center of the destination image.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

filter Type of resampling filter. It can be one of the following:

MLIB_NEAREST
MLIB_BILINEAR
MLIB_BICUBIC
MLIB_BICUBIC2

edge Type of edge condition. It can be one of the following:

MLIB_EDGE_DST_NO_WRITE
MLIB_EDGE_DST_FILL_ZERO
MLIB_EDGE_OP_NEAREST
MLIB_EDGE_SRC_EXTEND
MLIB_EDGE_SRC_PADDED

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_ImageZoom(3MLIB), mllib_ImageZoom_Fp(3MLIB), mllib_ImageZoomIn2X_Fp(3MLIB), mllib_ImageZoomIn2XIndex(3MLIB), mllib_ImageZoomIndex(3MLIB), mllib_ImageZoomOut2X(3MLIB), mllib_ImageZoomOut2X_Fp(3MLIB), mllib_ImageZoomOut2XIndex(3MLIB), attributes(5)

mllib_ImageZoomIn2X_Fp(3MLIB)

NAME	mllib_ImageZoomIn2X_Fp – 2X zoom
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageZoomIn2X_Fp(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, mllib_filter <i>filter</i>, mllib_edge <i>edge</i>);</pre>
DESCRIPTION	<p>The <code>mllib_ImageZoomIn2X_Fp()</code> function enlarges the floating-point source image by a factor of two. It uses the interpolation method as described by the resampling filter.</p> <p>The center of the upper-left corner pixel of an image is located at $(0.5, 0.5)$.</p> <p>The width and height of the destination image can be different from those of the source image.</p> <p>The center of the source image is mapped onto the center of the destination image.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>filter</i> Type of resampling filter. It can be one of the following:</p> <p style="padding-left: 40px;">MLIB_NEAREST MLIB_BILINEAR MLIB_BICUBIC MLIB_BICUBIC2</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <p style="padding-left: 40px;">MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_OP_NEAREST MLIB_EDGE_SRC_EXTEND MLIB_EDGE_SRC_PADDED</p>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

`mlib_ImageZoomIn2X_Fp(3MLIB)`

SEE ALSO `mlib_ImageZoom(3MLIB)`, `mlib_ImageZoom_Fp(3MLIB)`,
`mlib_ImageZoomIn2X(3MLIB)`, `mlib_ImageZoomIn2XIndex(3MLIB)`,
`mlib_ImageZoomIndex(3MLIB)`, `mlib_ImageZoomOut2X(3MLIB)`,
`mlib_ImageZoomOut2X_Fp(3MLIB)`, `mlib_ImageZoomOut2XIndex(3MLIB)`,
`attributes(5)`

mllib_ImageZoomIn2XIndex(3MLIB)

NAME	mllib_ImageZoomIn2XIndex – 2X zoom on color-indexed image
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageZoomIn2XIndex(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, mllib_filter <i>filter</i>, mllib_edge <i>edge</i>, const void *<i>colormap</i>);</pre>
DESCRIPTION	<p>The <code>mllib_ImageZoomIn2XIndex()</code> function enlarges the source image by a factor of two. It uses the interpolation method as described by the resampling filter.</p> <p>The image data type must be <code>MLIB_BYTE</code> or <code>MLIB_SHORT</code>.</p> <p>The center of the upper-left corner pixel of an image is located at $(0.5, 0.5)$.</p> <p>The width and height of the destination image can be different from those of the source image.</p> <p>The center of the source image is mapped onto the center of the destination image.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>filter</i> Type of resampling filter. It can be one of the following:</p> <pre>MLIB_NEAREST MLIB_BILINEAR MLIB_BICUBIC MLIB_BICUBIC2</pre> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_OP_NEAREST MLIB_EDGE_SRC_EXTEND MLIB_EDGE_SRC_PADDED</pre> <p><i>colormap</i> Internal data structure for inverse color mapping. This data structure is generated by the <code>mllib_ImageColorTrue2IndexInit()</code> function.</p>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

mllib_ImageZoomIn2XIndex(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	MT-Safe

SEE ALSO

mllib_ImageZoom(3MLIB), mllib_ImageZoom_Fp(3MLIB),
mllib_ImageZoomIn2X(3MLIB), mllib_ImageZoomIn2X_Fp(3MLIB),
mllib_ImageZoomIndex(3MLIB), mllib_ImageZoomOut2X(3MLIB),
mllib_ImageZoomOut2X_Fp(3MLIB), mllib_ImageZoomOut2XIndex(3MLIB),
attributes(5)

mllib_ImageZoomIndex(3MLIB)

NAME	mllib_ImageZoomIndex – zoom on color-indexed image														
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageZoomIndex(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, mllib_d64 <i>zoomx</i>, mllib_d64 <i>zoomy</i>, mllib_filter <i>filter</i>, mllib_edge <i>edge</i>, const void *<i>colormap</i>);</pre>														
DESCRIPTION	<p>The <code>mllib_ImageZoomIndex()</code> function will enlarge or minify the source image by the X and Y zoom factors. It uses the interpolation method as described by the resampling filter.</p> <p>The image data type must be <code>MLIB_BYTE</code> or <code>MLIB_SHORT</code>.</p> <p>The center of the upper-left corner pixel of an image is located at (0.5, 0.5).</p> <p>The width and height of the destination image can be different from those of the source image.</p> <p>The center of the source image is mapped onto the center of the destination image.</p>														
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>dst</i></td><td>Pointer to destination image.</td></tr><tr><td><i>src</i></td><td>Pointer to source image.</td></tr><tr><td><i>zoomx</i></td><td>X zoom factor. <i>zoomx</i> > 0.0.</td></tr><tr><td><i>zoomy</i></td><td>Y zoom factor. <i>zoomy</i> > 0.0.</td></tr><tr><td><i>filter</i></td><td>Type of resampling filter. It can be one of the following: <code>MLIB_NEAREST</code> <code>MLIB_BILINEAR</code> <code>MLIB_BICUBIC</code> <code>MLIB_BICUBIC2</code></td></tr><tr><td><i>edge</i></td><td>Type of edge condition. It can be one of the following: <code>MLIB_EDGE_DST_NO_WRITE</code> <code>MLIB_EDGE_DST_FILL_ZERO</code> <code>MLIB_EDGE_OP_NEAREST</code> <code>MLIB_EDGE_SRC_EXTEND</code> <code>MLIB_EDGE_SRC_PADDED</code></td></tr><tr><td><i>colormap</i></td><td>Internal data structure for inverse color mapping. This data structure is generated by the <code>mllib_ImageColorTrue2IndexInit()</code> function.</td></tr></table>	<i>dst</i>	Pointer to destination image.	<i>src</i>	Pointer to source image.	<i>zoomx</i>	X zoom factor. <i>zoomx</i> > 0.0.	<i>zoomy</i>	Y zoom factor. <i>zoomy</i> > 0.0.	<i>filter</i>	Type of resampling filter. It can be one of the following: <code>MLIB_NEAREST</code> <code>MLIB_BILINEAR</code> <code>MLIB_BICUBIC</code> <code>MLIB_BICUBIC2</code>	<i>edge</i>	Type of edge condition. It can be one of the following: <code>MLIB_EDGE_DST_NO_WRITE</code> <code>MLIB_EDGE_DST_FILL_ZERO</code> <code>MLIB_EDGE_OP_NEAREST</code> <code>MLIB_EDGE_SRC_EXTEND</code> <code>MLIB_EDGE_SRC_PADDED</code>	<i>colormap</i>	Internal data structure for inverse color mapping. This data structure is generated by the <code>mllib_ImageColorTrue2IndexInit()</code> function.
<i>dst</i>	Pointer to destination image.														
<i>src</i>	Pointer to source image.														
<i>zoomx</i>	X zoom factor. <i>zoomx</i> > 0.0.														
<i>zoomy</i>	Y zoom factor. <i>zoomy</i> > 0.0.														
<i>filter</i>	Type of resampling filter. It can be one of the following: <code>MLIB_NEAREST</code> <code>MLIB_BILINEAR</code> <code>MLIB_BICUBIC</code> <code>MLIB_BICUBIC2</code>														
<i>edge</i>	Type of edge condition. It can be one of the following: <code>MLIB_EDGE_DST_NO_WRITE</code> <code>MLIB_EDGE_DST_FILL_ZERO</code> <code>MLIB_EDGE_OP_NEAREST</code> <code>MLIB_EDGE_SRC_EXTEND</code> <code>MLIB_EDGE_SRC_PADDED</code>														
<i>colormap</i>	Internal data structure for inverse color mapping. This data structure is generated by the <code>mllib_ImageColorTrue2IndexInit()</code> function.														
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .														

mllib_ImageZoomIndex(3MLIB)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageZoom(3MLIB)`, `mllib_ImageZoom_Fp(3MLIB)`,
`mllib_ImageZoomIn2X(3MLIB)`, `mllib_ImageZoomIn2X_Fp(3MLIB)`,
`mllib_ImageZoomIn2XIndex(3MLIB)`, `mllib_ImageZoomOut2X(3MLIB)`,
`mllib_ImageZoomOut2X_Fp(3MLIB)`, `mllib_ImageZoomOut2XIndex(3MLIB)`,
`attributes(5)`

mllib_ImageZoomOut2X(3MLIB)

NAME	mllib_ImageZoomOut2X – 0.5X zoom						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageZoomOut2X(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, mllib_filter <i>filter</i>, mllib_edge <i>edge</i>);</pre>						
DESCRIPTION	<p>The <code>mllib_ImageZoomOut2X()</code> function minifies the source image by a factor of two. It uses the interpolation method as described by the resampling filter.</p> <p>The center of the upper-left corner pixel of an image is located at (0.5, 0.5).</p> <p>The width and height of the destination image can be different from those of the source image.</p> <p>The center of the source image is mapped onto the center of the destination image.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Pointer to destination image.</p> <p><i>src</i> Pointer to source image.</p> <p><i>filter</i> Type of resampling filter. It can be one of the following:</p> <p style="margin-left: 40px;">MLIB_NEAREST MLIB_BILINEAR MLIB_BICUBIC MLIB_BICUBIC2</p> <p><i>edge</i> Type of edge condition. It can be one of the following:</p> <p style="margin-left: 40px;">MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_OP_NEAREST MLIB_EDGE_SRC_EXTEND MLIB_EDGE_SRC_PADDED</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<p><code>mllib_ImageZoom(3MLIB)</code>, <code>mllib_ImageZoom_Fp(3MLIB)</code>, <code>mllib_ImageZoomIn2X(3MLIB)</code>, <code>mllib_ImageZoomIn2X_Fp(3MLIB)</code>, <code>mllib_ImageZoomIn2XIndex(3MLIB)</code>, <code>mllib_ImageZoomIndex(3MLIB)</code>, <code>mllib_ImageZoomOut2X_Fp(3MLIB)</code>, <code>mllib_ImageZoomOut2XIndex(3MLIB)</code>, <code>attributes(5)</code></p>						

NAME mllib_ImageZoomOut2X_Fp – 0.5X zoom

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageZoomOut2X_Fp(mllib_image *dst, const
    mllib_image *src, mllib_filter filter, mllib_edge edge);
```

DESCRIPTION The `mllib_ImageZoomOut2X_Fp()` function minifies the floating-point source image by a factor of two. It uses the interpolation method as described by the resampling filter.

The center of the upper-left corner pixel of an image is located at (0.5, 0.5).

The width and height of the destination image can be different from those of the source image.

The center of the source image is mapped onto the center of the destination image.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

filter Type of resampling filter. It can be one of the following:

MLIB_NEAREST
MLIB_BILINEAR
MLIB_BICUBIC
MLIB_BICUBIC2

edge Type of edge condition. It can be one of the following:

MLIB_EDGE_DST_NO_WRITE
MLIB_EDGE_DST_FILL_ZERO
MLIB_EDGE_OP_NEAREST
MLIB_EDGE_SRC_EXTEND
MLIB_EDGE_SRC_PADDED

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

`mlib_ImageZoomOut2X_Fp(3MLIB)`

SEE ALSO | `mlib_ImageZoom(3MLIB)`, `mlib_ImageZoom_Fp(3MLIB)`,
`mlib_ImageZoomIn2X(3MLIB)`, `mlib_ImageZoomIn2X_Fp(3MLIB)`,
`mlib_ImageZoomIn2XIndex(3MLIB)`, `mlib_ImageZoomIndex(3MLIB)`,
`mlib_ImageZoomOut2X(3MLIB)`, `mlib_ImageZoomOut2XIndex(3MLIB)`,
`attributes(5)`

NAME mllib_ImageZoomOut2XIndex – 0.5X zoom

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_ImageZoomOut2XIndex(mllib_image *dst, const
    mllib_image *src, mllib_filter filter, mllib_edge edge, const void
    *colormap);
```

DESCRIPTION The mllib_ImageZoomOut2XIndex() function minifies the source image by a factor of two. It uses the interpolation method as described by the resampling filter.

The image data type must be MLIB_BYTE or MLIB_SHORT.

The center of the upper-left corner pixel of an image is located at (0.5, 0.5).

The width and height of the destination image can be different from those of the source image.

The center of the source image is mapped onto the center of the destination image.

PARAMETERS The function takes the following arguments:

dst Pointer to destination image.

src Pointer to source image.

filter Type of resampling filter. It can be one of the following:

```
MLIB_NEAREST
MLIB_BILINEAR
MLIB_BICUBIC
MLIB_BICUBIC2
```

edge Type of edge condition. It can be one of the following:

```
MLIB_EDGE_DST_NO_WRITE
MLIB_EDGE_DST_FILL_ZERO
MLIB_EDGE_OP_NEAREST
MLIB_EDGE_SRC_EXTEND
MLIB_EDGE_SRC_PADDED
```

colormap Internal data structure for inverse color mapping. This data structure is generated by the mllib_ImageColorTrue2IndexInit() function.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

mllib_ImageZoomOut2XIndex(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	MT-Safe

SEE ALSO mllib_ImageZoom(3MLIB), mllib_ImageZoom_Fp(3MLIB),
mllib_ImageZoomIn2X(3MLIB), mllib_ImageZoomIn2X_Fp(3MLIB),
mllib_ImageZoomIn2XIndex(3MLIB), mllib_ImageZoomIndex(3MLIB),
mllib_ImageZoomOut2X(3MLIB), mllib_ImageZoomOut2X_Fp(3MLIB),
attributes(5)

NAME	mllib_ImageZoomTranslate – zoom, with translation																
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageZoomTranslate(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, mllib_d64 <i>zoomx</i>, mllib_d64 <i>zoomy</i>, mllib_d64 <i>tx</i>, mllib_d64 <i>ty</i>, mllib_filter <i>filter</i>, mllib_edge <i>edge</i>);</pre>																
DESCRIPTION	<p>The <code>mllib_ImageZoomTranslate()</code> function will enlarge or minify the source image by the X and Y zoom factors, with translation. It uses the interpolation method as described by the resampling filter.</p> <p>It uses the following equation for coordinate mapping:</p> <pre>xd = zoomx*xs + tx yd = zoomy*ys + ty</pre> <p>where a point with coordinates (<i>xs</i>, <i>ys</i>) in the source image is mapped to a point with coordinates (<i>xd</i>, <i>yd</i>) in the destination image.</p> <p>The data type of the images can be <code>MLIB_BIT</code>, <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p> <p>The width and height of the destination image can be different from the width and height of the source image.</p> <p>The center of the upper-left corner pixel of an image is located at (0.5, 0.5).</p>																
PARAMETERS	<p>The function takes the following arguments:</p> <table border="0"> <tr> <td style="padding-right: 20px;"><i>dst</i></td> <td>Pointer to destination image.</td> </tr> <tr> <td><i>src</i></td> <td>Pointer to source image.</td> </tr> <tr> <td><i>zoomx</i></td> <td>X zoom factor. <code>zoomx > 0</code>.</td> </tr> <tr> <td><i>zoomy</i></td> <td>Y zoom factor. <code>zoomy > 0</code>.</td> </tr> <tr> <td><i>tx</i></td> <td>X translation.</td> </tr> <tr> <td><i>ty</i></td> <td>Y translation.</td> </tr> <tr> <td><i>filter</i></td> <td>Type of resampling filter. It can be one of the following: <ul style="list-style-type: none"> <code>MLIB_NEAREST</code> <code>MLIB_BILINEAR</code> <code>MLIB_BICUBIC</code> <code>MLIB_BICUBIC2</code> </td> </tr> <tr> <td><i>edge</i></td> <td>Type of edge condition. It can be one of the following: <ul style="list-style-type: none"> <code>MLIB_EDGE_DST_NO_WRITE</code> <code>MLIB_EDGE_DST_FILL_ZERO</code> <code>MLIB_EDGE_OP_NEAREST</code> <code>MLIB_EDGE_SRC_EXTEND</code> <code>MLIB_EDGE_SRC_EXTEND_INDEF</code> <code>MLIB_EDGE_SRC_PADDED</code> </td> </tr> </table>	<i>dst</i>	Pointer to destination image.	<i>src</i>	Pointer to source image.	<i>zoomx</i>	X zoom factor. <code>zoomx > 0</code> .	<i>zoomy</i>	Y zoom factor. <code>zoomy > 0</code> .	<i>tx</i>	X translation.	<i>ty</i>	Y translation.	<i>filter</i>	Type of resampling filter. It can be one of the following: <ul style="list-style-type: none"> <code>MLIB_NEAREST</code> <code>MLIB_BILINEAR</code> <code>MLIB_BICUBIC</code> <code>MLIB_BICUBIC2</code> 	<i>edge</i>	Type of edge condition. It can be one of the following: <ul style="list-style-type: none"> <code>MLIB_EDGE_DST_NO_WRITE</code> <code>MLIB_EDGE_DST_FILL_ZERO</code> <code>MLIB_EDGE_OP_NEAREST</code> <code>MLIB_EDGE_SRC_EXTEND</code> <code>MLIB_EDGE_SRC_EXTEND_INDEF</code> <code>MLIB_EDGE_SRC_PADDED</code>
<i>dst</i>	Pointer to destination image.																
<i>src</i>	Pointer to source image.																
<i>zoomx</i>	X zoom factor. <code>zoomx > 0</code> .																
<i>zoomy</i>	Y zoom factor. <code>zoomy > 0</code> .																
<i>tx</i>	X translation.																
<i>ty</i>	Y translation.																
<i>filter</i>	Type of resampling filter. It can be one of the following: <ul style="list-style-type: none"> <code>MLIB_NEAREST</code> <code>MLIB_BILINEAR</code> <code>MLIB_BICUBIC</code> <code>MLIB_BICUBIC2</code> 																
<i>edge</i>	Type of edge condition. It can be one of the following: <ul style="list-style-type: none"> <code>MLIB_EDGE_DST_NO_WRITE</code> <code>MLIB_EDGE_DST_FILL_ZERO</code> <code>MLIB_EDGE_OP_NEAREST</code> <code>MLIB_EDGE_SRC_EXTEND</code> <code>MLIB_EDGE_SRC_EXTEND_INDEF</code> <code>MLIB_EDGE_SRC_PADDED</code> 																

mllib_ImageZoomTranslate(3MLIB)

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageZoomTranslate_Fp(3MLIB)`, `mllib_ImageAffine(3MLIB)`, `mllib_ImageAffine_Fp(3MLIB)`, `attributes(5)`

NAME	mllib_ImageZoomTranslateBlend – image scaling with alpha blending
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageZoomTranslateBlend(mllib_image *dst, const mllib_image *src, mllib_d64 zoomx, mllib_d64 zoomy, mllib_d64 tx, mllib_d64 ty, mllib_filter filter, mllib_edge edge, mllib_blend blend, mllib_s32 alpha, mllib_s32 cmask);</pre>
DESCRIPTION	<p>The <code>mllib_ImageZoomTranslateBlend()</code> function will enlarge or minify the source image by the X and Y zoom factors, with translation, and blend it with the destination image.</p> <p>It uses the following equation for coordinate mapping:</p> <pre>xd = zoomx*xs + tx yd = zoomy*ys + ty</pre> <p>where a point with coordinates (<i>xs</i>, <i>ys</i>) in the source image is mapped to a point with coordinates (<i>xd</i>, <i>yd</i>) in the destination image.</p> <p>The center of the upper-left corner pixel of an image is located at (0.5, 0.5).</p> <p>The alpha blending is closely combined with the interpolation to achieve better performance. Part of alpha blending has to be performed before or together with the interpolation if the source image has an alpha channel. In that case, the color components of each neighboring source pixel which participates in the interpolation (<i>src_r</i> and etc.) have to be pre-multiplied by the alpha component of the same source pixel (<i>src_a</i>). After the interpolation, the overall alpha (<i>alpha</i>), the interpolated source alpha (<i>interp_a</i>) and the destination pixel's original alpha (<i>dst_a</i>, if any) are used to blend the interpolated source pixel (with components <i>interp_r</i> and etc.) with the destination pixel (with components <i>dst_r</i> and etc.).</p> <p>The <code>MLIB_BLEND_GTK_SRC</code> blending is similar to the <code>SRC</code> rule of the Porter-Duff rules for image compositing. It is defined by</p> <pre>Cd = Cs Ad = As</pre> <p>in general, and by the following formula for this function:</p> <pre>if (interp_a != 0.0) { if (dst_has_alpha) { dst_r = interp_r/interp_a; dst_g = interp_g/interp_a; dst_b = interp_b/interp_a; dst_a = interp_a; } else { dst_r = interp_r; dst_g = interp_g; dst_b = interp_b; dst_a = 1.0; // implied } } else {</pre>

mllib_ImageZoomTranslateBlend(3MLIB)

```
    dst_r = 0;
    dst_g = 0;
    dst_b = 0;
    dst_a = 0;
}
```

The `MLIB_BLEND_GTK_SRC_OVER` or `MLIB_BLEND_GTK_SRC_OVER2` blending is similar to the `SRC_OVER` rule of the Porter-Duff rules for image compositing. It is defined by

```
Cd = Cs + Cd*(1 - As)
Ad = As + Ad*(1 - As)
```

in general, and by the following formula for this function:

```
w = alpha*interp_a + (1 - alpha*interp_a)*dst_a;
if (w != 0.0) {
    dst_r = (alpha*interp_r +
            (1 - alpha*interp_a)*dst_a*dst_r)/w;
    dst_g = (alpha*interp_g +
            (1 - alpha*interp_a)*dst_a*dst_g)/w;
    dst_b = (alpha*interp_b +
            (1 - alpha*interp_a)*dst_a*dst_b)/w;
    dst_a = w;
} else if (MLIB_BLEND_GTK_SRC_OVER) {
    dst_r = 0;
    dst_g = 0;
    dst_b = 0;
    dst_a = 0;
}
```

where *alpha*, *src_a*, *interp_a* and *dst_a* are assumed to be in the range of [0.0, 1.0].

For an image with 4 channels, the first or the fourth channel is considered the alpha channel if *cmask* equals 8 or 1, respectively. An image with 3 channels is considered to have no alpha channel, which is equivalent to having an alpha channel filled with all 1.0, or 0xff in case of `MLIB_BYTE`, if the general formulas for blending shown above are used.

Both *src* and *dst* must be of type `MLIB_BYTE`. They can have either 3 or 4 channels.

The *src* image cannot have width or height larger than 32767.

PARAMETERS

The function takes the following arguments:

<i>dst</i>	Pointer to destination image.
<i>src</i>	Pointer to first source image.
<i>zoomx</i>	X zoom factor. <i>zoomx</i> > 0.0.
<i>zoomy</i>	Y zoom factor. <i>zoomy</i> > 0.0.
<i>tx</i>	X translation.
<i>ty</i>	Y translation.

mllib_ImageZoomTranslateBlend(3MLIB)

filter Type of resampling filter. It can be one of the following:
 MLIB_NEAREST
 MLIB_BILINEAR
 MLIB_BICUBIC
 MLIB_BICUBIC2

edge Type of edge condition. It can be one of the following:
 MLIB_EDGE_DST_NO_WRITE
 MLIB_EDGE_DST_FILL_ZERO
 MLIB_EDGE_OP_NEAREST
 MLIB_EDGE_SRC_EXTEND
 MLIB_EDGE_SRC_EXTEND_INDEF
 MLIB_EDGE_SRC_PADDED

blend Type of alpha blending. It can be one of the following:
 MLIB_BLEND_GTK_SRC
 MLIB_BLEND_GTK_SRC_OVER
 MLIB_BLEND_GTK_SRC_OVER2

alpha Overall alpha for blending.

cmask Channel mask to indicate the alpha channel.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_ImageZoomBlend\(3MLIB\)](#),
[mllib_ImageZoomTranslateTableBlend\(3MLIB\)](#), [attributes\(5\)](#)

mllib_ImageZoomTranslate_Fp(3MLIB)

NAME	mllib_ImageZoomTranslate_Fp – zoom, with translation																
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageZoomTranslate_Fp(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, mllib_d64 <i>zoomx</i>, mllib_d64 <i>zoomy</i>, mllib_d64 <i>tx</i>, mllib_d64 <i>ty</i>, mllib_filter <i>filter</i>, mllib_edge <i>edge</i>);</pre>																
DESCRIPTION	<p>The <code>mllib_ImageZoomTranslate_Fp()</code> function will enlarge or minify the floating-point source image by the X and Y zoom factors, with translation. It uses the interpolation method as described by the resampling filter.</p> <p>It uses the following equation for coordinate mapping:</p> $\begin{aligned}x_d &= \text{zoomx} * x_s + t_x \\ y_d &= \text{zoomy} * y_s + t_y\end{aligned}$ <p>where a point with coordinates (x_s, y_s) in the source image is mapped to a point with coordinates (x_d, y_d) in the destination image.</p> <p>The data type of the images can be <code>MLIB_FLOAT</code> or <code>MLIB_DOUBLE</code>.</p> <p>The width and height of the destination image can be different from the width and height of the source image.</p> <p>The center of the upper-left corner pixel of an image is located at (0.5, 0.5).</p>																
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>dst</i></td><td>Pointer to destination image.</td></tr><tr><td><i>src</i></td><td>Pointer to source image.</td></tr><tr><td><i>zoomx</i></td><td>X zoom factor. $\text{zoomx} > 0$.</td></tr><tr><td><i>zoomy</i></td><td>Y zoom factor. $\text{zoomy} > 0$.</td></tr><tr><td><i>tx</i></td><td>X translation.</td></tr><tr><td><i>ty</i></td><td>Y translation.</td></tr><tr><td><i>filter</i></td><td>Type of resampling filter. It can be one of the following: <code>MLIB_NEAREST</code> <code>MLIB_BILINEAR</code> <code>MLIB_BICUBIC</code> <code>MLIB_BICUBIC2</code></td></tr><tr><td><i>edge</i></td><td>Type of edge condition. It can be one of the following: <code>MLIB_EDGE_DST_NO_WRITE</code> <code>MLIB_EDGE_DST_FILL_ZERO</code> <code>MLIB_EDGE_OP_NEAREST</code> <code>MLIB_EDGE_SRC_EXTEND</code> <code>MLIB_EDGE_SRC_EXTEND_INDEF</code> <code>MLIB_EDGE_SRC_PADDED</code></td></tr></table>	<i>dst</i>	Pointer to destination image.	<i>src</i>	Pointer to source image.	<i>zoomx</i>	X zoom factor. $\text{zoomx} > 0$.	<i>zoomy</i>	Y zoom factor. $\text{zoomy} > 0$.	<i>tx</i>	X translation.	<i>ty</i>	Y translation.	<i>filter</i>	Type of resampling filter. It can be one of the following: <code>MLIB_NEAREST</code> <code>MLIB_BILINEAR</code> <code>MLIB_BICUBIC</code> <code>MLIB_BICUBIC2</code>	<i>edge</i>	Type of edge condition. It can be one of the following: <code>MLIB_EDGE_DST_NO_WRITE</code> <code>MLIB_EDGE_DST_FILL_ZERO</code> <code>MLIB_EDGE_OP_NEAREST</code> <code>MLIB_EDGE_SRC_EXTEND</code> <code>MLIB_EDGE_SRC_EXTEND_INDEF</code> <code>MLIB_EDGE_SRC_PADDED</code>
<i>dst</i>	Pointer to destination image.																
<i>src</i>	Pointer to source image.																
<i>zoomx</i>	X zoom factor. $\text{zoomx} > 0$.																
<i>zoomy</i>	Y zoom factor. $\text{zoomy} > 0$.																
<i>tx</i>	X translation.																
<i>ty</i>	Y translation.																
<i>filter</i>	Type of resampling filter. It can be one of the following: <code>MLIB_NEAREST</code> <code>MLIB_BILINEAR</code> <code>MLIB_BICUBIC</code> <code>MLIB_BICUBIC2</code>																
<i>edge</i>	Type of edge condition. It can be one of the following: <code>MLIB_EDGE_DST_NO_WRITE</code> <code>MLIB_EDGE_DST_FILL_ZERO</code> <code>MLIB_EDGE_OP_NEAREST</code> <code>MLIB_EDGE_SRC_EXTEND</code> <code>MLIB_EDGE_SRC_EXTEND_INDEF</code> <code>MLIB_EDGE_SRC_PADDED</code>																

`mllib_ImageZoomTranslate_Fp(3MLIB)`

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageZoomTranslate(3MLIB)`, `mllib_ImageAffine(3MLIB)`, `mllib_ImageAffine_Fp(3MLIB)`, `attributes(5)`

mllib_ImageZoomTranslateTable(3MLIB)

NAME	mllib_ImageZoomTranslateTable – zoom, with translation, with table-driven interpolation																
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageZoomTranslateTable(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, mllib_d64 <i>zoomx</i>, mllib_d64 <i>zoomy</i>, mllib_d64 <i>tx</i>, mllib_d64 <i>ty</i>, const void *<i>interp_table</i>, mllib_edge <i>edge</i>);</pre>																
DESCRIPTION	<p>The <code>mllib_ImageZoomTranslateTable()</code> function will enlarge or minify the source image by the X and Y zoom factors, with translation. It uses a table, <code>interp_table</code>, to do interpolation.</p> <p>It uses the following equation for coordinate mapping:</p> $\begin{aligned}x_d &= \text{zoomx} * x_s + t_x \\ y_d &= \text{zoomy} * y_s + t_y\end{aligned}$ <p>where a point with coordinates (x_s, y_s) in the source image is mapped to a point with coordinates (x_d, y_d) in the destination image.</p> <p>The data type of the images can be <code>MLIB_BYTE</code>, <code>MLIB_SHORT</code>, <code>MLIB_USHORT</code>, or <code>MLIB_INT</code>.</p> <p>The width and height of the destination image can be different from the width and height of the source image.</p> <p>The center of the upper-left corner pixel of an image is located at (0.5, 0.5).</p>																
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>dst</i></td><td>Pointer to destination image.</td></tr><tr><td><i>src</i></td><td>Pointer to source image.</td></tr><tr><td><i>zoomx</i></td><td>X zoom factor. <code>zoomx > 0</code>.</td></tr><tr><td><i>zoomy</i></td><td>Y zoom factor. <code>zoomy > 0</code>.</td></tr><tr><td><i>tx</i></td><td>X translation.</td></tr><tr><td><i>ty</i></td><td>Y translation.</td></tr><tr><td><i>interp_table</i></td><td>Pointer to an interpolation table. The table is created by the <code>mllib_ImageInterpTableCreate()</code> function.</td></tr><tr><td><i>edge</i></td><td>Type of edge condition. It can be one of the following:</td></tr></table> <pre>MLIB_EDGE_DST_NO_WRITE MLIB_EDGE_DST_FILL_ZERO MLIB_EDGE_OP_NEAREST MLIB_EDGE_SRC_EXTEND MLIB_EDGE_SRC_EXTEND_INDEF MLIB_EDGE_SRC_PADDED</pre>	<i>dst</i>	Pointer to destination image.	<i>src</i>	Pointer to source image.	<i>zoomx</i>	X zoom factor. <code>zoomx > 0</code> .	<i>zoomy</i>	Y zoom factor. <code>zoomy > 0</code> .	<i>tx</i>	X translation.	<i>ty</i>	Y translation.	<i>interp_table</i>	Pointer to an interpolation table. The table is created by the <code>mllib_ImageInterpTableCreate()</code> function.	<i>edge</i>	Type of edge condition. It can be one of the following:
<i>dst</i>	Pointer to destination image.																
<i>src</i>	Pointer to source image.																
<i>zoomx</i>	X zoom factor. <code>zoomx > 0</code> .																
<i>zoomy</i>	Y zoom factor. <code>zoomy > 0</code> .																
<i>tx</i>	X translation.																
<i>ty</i>	Y translation.																
<i>interp_table</i>	Pointer to an interpolation table. The table is created by the <code>mllib_ImageInterpTableCreate()</code> function.																
<i>edge</i>	Type of edge condition. It can be one of the following:																

`mllib_ImageZoomTranslateTable(3MLIB)`

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageInterpTableCreate(3MLIB)`,
`mllib_ImageInterpTableDelete(3MLIB)`,
`mllib_ImageZoomTranslateTable_Fp(3MLIB)`,
`mllib_ImageZoomTranslate(3MLIB)`, `mllib_ImageZoomTranslate_Fp(3MLIB)`,
`attributes(5)`

mllib_ImageZoomTranslateTableBlend(3MLIB)

NAME	mllib_ImageZoomTranslateTableBlend – image scaling using interpolation table, combined with alpha blending
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageZoomTranslateTableBlend(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, mllib_d64 <i>zoomx</i>, mllib_d64 <i>zoomy</i>, mllib_d64 <i>tx</i>, mllib_d64 <i>ty</i>, const void *<i>table</i>, mllib_edge <i>edge</i>, mllib_blend <i>blend</i>, mllib_s32 <i>cmask</i>);</pre>
DESCRIPTION	<p>The <code>mllib_ImageZoomTranslateTableBlend()</code> function will enlarge or minify the source image by the X and Y zoom factors, with translation, and blend it with the destination image.</p> <p>It uses the following equation for coordinate mapping:</p> <pre>xd = zoomx*xs + tx yd = zoomy*ys + ty</pre> <p>where a point with coordinates (<i>xs</i>, <i>ys</i>) in the source image is mapped to a point with coordinates (<i>xd</i>, <i>yd</i>) in the destination image.</p> <p>The center of the upper-left corner pixel of an image is located at (0.5, 0.5).</p> <p>It is assumed that the overall alpha for controlling the blending between the source image and the destination image has been pre-multiplied to the interpolation table for better performance.</p> <p>The alpha blending is closely combined with the interpolation to achieve better performance. Part of alpha blending has to be performed before or together with the interpolation if the source image has an alpha channel. In that case, the color components of each neighboring source pixel which participates in the interpolation (<i>src_r</i> and etc.) have to be pre-multiplied by the alpha component of the same source pixel (<i>src_a</i>). After the interpolation, the interpolated alpha (<i>interp_a</i>, which has been multiplied by the overall alpha because of the pre-multiplied interpolation table) and the destination pixel's original alpha (<i>dst_a</i>, if any) are used to blend the interpolated source pixel (with components <i>interp_r</i> and etc.) with the destination pixel (with components <i>dst_r</i> and etc.).</p> <p>The <code>MLIB_BLEND_GTK_SRC</code> blending is similar to the <code>SRC</code> rule of the Porter-Duff rules for image compositing. It is defined by</p> <pre>Cd = Cs Ad = As</pre> <p>in general, and by the following formula for this function:</p> <pre>if (interp_a != 0.0) { if (dst_has_alpha) { dst_r = interp_r/interp_a; dst_g = interp_g/interp_a; dst_b = interp_b/interp_a; dst_a = interp_a; }</pre>


```

    } else {
        dst_r = interp_r;
        dst_g = interp_g;
        dst_b = interp_b;
        dst_a = 1.0; // implied
    }
} else {
    dst_r = 0;
    dst_g = 0;
    dst_b = 0;
    dst_a = 0;
}

```

The `MLIB_BLEND_GTK_SRC_OVER` or `MLIB_BLEND_GTK_SRC_OVER2` blending is similar to the `SRC_OVER` rule of the Porter-Duff rules for image compositing. It is defined by

$$C_d = C_s + C_d \cdot (1 - A_s)$$

$$A_d = A_s + A_d \cdot (1 - A_s)$$

in general, and by the following formula for this function:

```

w = interp_a + (1 - interp_a)*dst_a;
if (w != 0.0) {
    dst_r = (interp_r + (1 - interp_a)*dst_a*dst_r)/w;
    dst_g = (interp_g + (1 - interp_a)*dst_a*dst_g)/w;
    dst_b = (interp_b + (1 - interp_a)*dst_a*dst_b)/w;
    dst_a = w;
} else if (MLIB_BLEND_GTK_SRC_OVER) {
    dst_r = 0;
    dst_g = 0;
    dst_b = 0;
    dst_a = 0;
}

```

where `src_a`, `interp_a` and `dst_a` are assumed to be in the range of `[0.0, 1.0]`.

For an image with 4 channels, the first or the fourth channel is considered the alpha channel if `cmask` equals 8 or 1, respectively. An image with 3 channels is considered to have no alpha channel, which is equivalent to having an alpha channel filled with all 1.0, or `0xff` in case of `MLIB_BYTE`, if the general formulas for blending shown above are used.

Both `src` and `dst` must be of type `MLIB_BYTE`. They can have either 3 or 4 channels.

The `src` image cannot have width or height larger than 32767.

PARAMETERS

The function takes the following arguments:

<i>dst</i>	Pointer to destination image.
<i>src</i>	Pointer to first source image.
<i>zoomx</i>	X zoom factor. <code>zoomx > 0.0</code> .
<i>zoomy</i>	Y zoom factor. <code>zoomy > 0.0</code> .
<i>tx</i>	X translation.

mllib_ImageZoomTranslateTableBlend(3MLIB)

ty Y translation.

table Pointer to interpolation table structure.

edge Type of edge condition. It can be one of the following:
MLIB_EDGE_DST_NO_WRITE
MLIB_EDGE_DST_FILL_ZERO
MLIB_EDGE_OP_NEAREST
MLIB_EDGE_SRC_EXTEND
MLIB_EDGE_SRC_EXTEND_INDEF
MLIB_EDGE_SRC_PADDED

blend Type of alpha blending. It can be one of the following:
MLIB_BLEND_GTK_SRC
MLIB_BLEND_GTK_SRC_OVER
MLIB_BLEND_GTK_SRC_OVER2

cmask Channel mask to indicate the alpha channel.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageZoomBlend(3MLIB)`, `mllib_ImageZoomTranslateBlend(3MLIB)`, `mllib_ImageInterpTableCreate(3MLIB)`, `attributes(5)`

mllib_ImageZoomTranslateTable_Fp(3MLIB)

NAME	mllib_ImageZoomTranslateTable_Fp – zoom, with translation, with table-driven interpolation																
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_ImageZoomTranslateTable_Fp(mllib_image *<i>dst</i>, const mllib_image *<i>src</i>, mllib_d64 <i>zoomx</i>, mllib_d64 <i>zoomy</i>, mllib_d64 <i>tx</i>, mllib_d64 <i>ty</i>, const void *<i>interp_table</i>, mllib_edge <i>edge</i>);</pre>																
DESCRIPTION	<p>The <code>mllib_ImageZoomTranslateTable_Fp()</code> function will enlarge or minify the floating-point source image by the X and Y zoom factors, with translation. It uses a table, <code>interp_table</code>, to do interpolation.</p> <p>It uses the following equation for coordinate mapping:</p> $\begin{aligned}x_d &= \text{zoomx} * x_s + t_x \\ y_d &= \text{zoomy} * y_s + t_y\end{aligned}$ <p>where a point with coordinates (x_s, y_s) in the source image is mapped to a point with coordinates (x_d, y_d) in the destination image.</p> <p>The data type of the images can be <code>MLIB_FLOAT</code> or <code>MLIB_DOUBLE</code>.</p> <p>The width and height of the destination image can be different from the width and height of the source image.</p> <p>The center of the upper-left corner pixel of an image is located at (0.5, 0.5).</p>																
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>dst</i></td><td>Pointer to destination image.</td></tr><tr><td><i>src</i></td><td>Pointer to source image.</td></tr><tr><td><i>zoomx</i></td><td>X zoom factor. <code>zoomx > 0</code>.</td></tr><tr><td><i>zoomy</i></td><td>Y zoom factor. <code>zoomy > 0</code>.</td></tr><tr><td><i>tx</i></td><td>X translation.</td></tr><tr><td><i>ty</i></td><td>Y translation.</td></tr><tr><td><i>interp_table</i></td><td>Pointer to an interpolation table. The table is created by the <code>mllib_ImageInterpTableCreate()</code> function.</td></tr><tr><td><i>edge</i></td><td>Type of edge condition. It can be one of the following: <code>MLIB_EDGE_DST_NO_WRITE</code> <code>MLIB_EDGE_DST_FILL_ZERO</code> <code>MLIB_EDGE_OP_NEAREST</code> <code>MLIB_EDGE_SRC_EXTEND</code> <code>MLIB_EDGE_SRC_EXTEND_INDEF</code> <code>MLIB_EDGE_SRC_PADDED</code></td></tr></table>	<i>dst</i>	Pointer to destination image.	<i>src</i>	Pointer to source image.	<i>zoomx</i>	X zoom factor. <code>zoomx > 0</code> .	<i>zoomy</i>	Y zoom factor. <code>zoomy > 0</code> .	<i>tx</i>	X translation.	<i>ty</i>	Y translation.	<i>interp_table</i>	Pointer to an interpolation table. The table is created by the <code>mllib_ImageInterpTableCreate()</code> function.	<i>edge</i>	Type of edge condition. It can be one of the following: <code>MLIB_EDGE_DST_NO_WRITE</code> <code>MLIB_EDGE_DST_FILL_ZERO</code> <code>MLIB_EDGE_OP_NEAREST</code> <code>MLIB_EDGE_SRC_EXTEND</code> <code>MLIB_EDGE_SRC_EXTEND_INDEF</code> <code>MLIB_EDGE_SRC_PADDED</code>
<i>dst</i>	Pointer to destination image.																
<i>src</i>	Pointer to source image.																
<i>zoomx</i>	X zoom factor. <code>zoomx > 0</code> .																
<i>zoomy</i>	Y zoom factor. <code>zoomy > 0</code> .																
<i>tx</i>	X translation.																
<i>ty</i>	Y translation.																
<i>interp_table</i>	Pointer to an interpolation table. The table is created by the <code>mllib_ImageInterpTableCreate()</code> function.																
<i>edge</i>	Type of edge condition. It can be one of the following: <code>MLIB_EDGE_DST_NO_WRITE</code> <code>MLIB_EDGE_DST_FILL_ZERO</code> <code>MLIB_EDGE_OP_NEAREST</code> <code>MLIB_EDGE_SRC_EXTEND</code> <code>MLIB_EDGE_SRC_EXTEND_INDEF</code> <code>MLIB_EDGE_SRC_PADDED</code>																
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .																

mllib_ImageZoomTranslateTable_Fp(3MLIB)

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_ImageInterpTableCreate(3MLIB)`,
`mllib_ImageInterpTableDelete(3MLIB)`,
`mllib_ImageZoomTranslateTable(3MLIB)`,
`mllib_ImageZoomTranslate(3MLIB)`, `mllib_ImageZoomTranslate_Fp(3MLIB)`,
`attributes(5)`

mllib_ImageZoomTranslateToGray(3MLIB)

NAME	mllib_ImageZoomTranslateToGray – zoom, with translation, and convert to grayscale																				
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_ImageZoomTranslateToGray(mllib_image *dst, const mllib_image *src, mllib_d64 zoomx, mllib_d64 zoomy, mllib_d64 tx, mllib_d64 ty, mllib_filter filter, mllib_edge edge, const mllib_s32 *ghigh, const mllib_s32 *glow);</pre>																				
DESCRIPTION	<p>The <code>mllib_ImageZoomTranslateToGray()</code> function will enlarge or minify the source binary image by the X and Y zoom factors, with translation, and convert the resulting image into a grayscale image.</p> <p>It uses the following equation for coordinate mapping:</p> <pre>xd = zoomx*xs + tx yd = zoomy*ys + ty</pre> <p>where a point with coordinates (<i>xs</i>, <i>ys</i>) in the source image is mapped to a point with coordinates (<i>xd</i>, <i>yd</i>) in the destination image.</p> <p>The width and height of the destination image can be different from the width and height of the source image.</p> <p>The center of the upper-left corner pixel of an image is located at (0.5, 0.5).</p>																				
PARAMETERS	<p>The function takes the following arguments:</p> <table border="0"> <tr> <td style="padding-right: 20px;"><i>dst</i></td> <td>Pointer to destination image. It must be of type <code>MLIB_BYTE</code> and have just one channel.</td> </tr> <tr> <td><i>src</i></td> <td>Pointer to source image. It must be of type <code>MLIB_BIT</code> and have just one channel.</td> </tr> <tr> <td><i>zoomx</i></td> <td>X zoom factor. <code>zoomx > 0</code>.</td> </tr> <tr> <td><i>zoomy</i></td> <td>Y zoom factor. <code>zoomy > 0</code>.</td> </tr> <tr> <td><i>tx</i></td> <td>X translation.</td> </tr> <tr> <td><i>ty</i></td> <td>Y translation.</td> </tr> <tr> <td><i>filter</i></td> <td>Type of resampling filter. It must be <code>MLIB_NEAREST</code>.</td> </tr> <tr> <td><i>edge</i></td> <td>Type of edge condition. It can be one of the following: <code>MLIB_EDGE_DST_NO_WRITE</code> <code>MLIB_EDGE_DST_FILL_ZERO</code> <code>MLIB_EDGE_OP_NEAREST</code> <code>MLIB_EDGE_SRC_EXTEND</code> <code>MLIB_EDGE_SRC_PADDED</code> </td> </tr> <tr> <td><i>ghigh</i></td> <td>Pointer to value for '1' pixels in source image.</td> </tr> <tr> <td><i>glow</i></td> <td>Pointer to value for '0' pixels in source image.</td> </tr> </table>	<i>dst</i>	Pointer to destination image. It must be of type <code>MLIB_BYTE</code> and have just one channel.	<i>src</i>	Pointer to source image. It must be of type <code>MLIB_BIT</code> and have just one channel.	<i>zoomx</i>	X zoom factor. <code>zoomx > 0</code> .	<i>zoomy</i>	Y zoom factor. <code>zoomy > 0</code> .	<i>tx</i>	X translation.	<i>ty</i>	Y translation.	<i>filter</i>	Type of resampling filter. It must be <code>MLIB_NEAREST</code> .	<i>edge</i>	Type of edge condition. It can be one of the following: <code>MLIB_EDGE_DST_NO_WRITE</code> <code>MLIB_EDGE_DST_FILL_ZERO</code> <code>MLIB_EDGE_OP_NEAREST</code> <code>MLIB_EDGE_SRC_EXTEND</code> <code>MLIB_EDGE_SRC_PADDED</code>	<i>ghigh</i>	Pointer to value for '1' pixels in source image.	<i>glow</i>	Pointer to value for '0' pixels in source image.
<i>dst</i>	Pointer to destination image. It must be of type <code>MLIB_BYTE</code> and have just one channel.																				
<i>src</i>	Pointer to source image. It must be of type <code>MLIB_BIT</code> and have just one channel.																				
<i>zoomx</i>	X zoom factor. <code>zoomx > 0</code> .																				
<i>zoomy</i>	Y zoom factor. <code>zoomy > 0</code> .																				
<i>tx</i>	X translation.																				
<i>ty</i>	Y translation.																				
<i>filter</i>	Type of resampling filter. It must be <code>MLIB_NEAREST</code> .																				
<i>edge</i>	Type of edge condition. It can be one of the following: <code>MLIB_EDGE_DST_NO_WRITE</code> <code>MLIB_EDGE_DST_FILL_ZERO</code> <code>MLIB_EDGE_OP_NEAREST</code> <code>MLIB_EDGE_SRC_EXTEND</code> <code>MLIB_EDGE_SRC_PADDED</code>																				
<i>ghigh</i>	Pointer to value for '1' pixels in source image.																				
<i>glow</i>	Pointer to value for '0' pixels in source image.																				

`mllib_ImageZoomTranslateToGray(3MLIB)`

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_ImageSubsampleBinaryToGray(3MLIB)`, `attributes(5)`

Multimedia Library Functions Part Two

mllib_malloc(3MLIB)

NAME mllib_malloc – allocate a block of bytes

SYNOPSIS cc [*flag...*] *file...* -lmllib [*library...*]
#include <mllib.h>

```
void *mllib_malloc(size_t size);
```

DESCRIPTION The mllib_malloc() function allocates *size* bytes on an 8-byte aligned boundary and returns a pointer to the allocated block.

This function is equivalent to memalign(8, size).

PARAMETERS The function takes the following arguments:

size Size of the block in bytes.

RETURN VALUES The function returns a pointer to the allocated block if successful. Otherwise it returns a null pointer.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_free(3MLIB), mllib_realloc(3MLIB), malloc(3C), attributes(5)

mllib_MatrixAddS_U8_Mod(3MLIB)

NAME | mllib_MatrixAddS_U8_Mod, mllib_MatrixAddS_U8_Sat, mllib_MatrixAddS_U8C_Mod, mllib_MatrixAddS_U8C_Sat, mllib_MatrixAddS_S8_Mod, mllib_MatrixAddS_S8_Sat, mllib_MatrixAddS_S8C_Mod, mllib_MatrixAddS_S8C_Sat, mllib_MatrixAddS_S16_Mod, mllib_MatrixAddS_S16_Sat, mllib_MatrixAddS_S16C_Mod, mllib_MatrixAddS_S16C_Sat, mllib_MatrixAddS_S32_Mod, mllib_MatrixAddS_S32_Sat, mllib_MatrixAddS_S32C_Mod, mllib_MatrixAddS_S32C_Sat – matrix addition to scalar, in place

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_MatrixAddS_U8_Mod(mllib_u8 *xz, const mllib_u8 *c,
    mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixAddS_U8_Sat(mllib_u8 *xz, const mllib_u8 *c,
    mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixAddS_U8C_Mod(mllib_u8 *xz, const mllib_u8
    *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixAddS_U8C_Sat(mllib_u8 *xz, const mllib_u8
    *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixAddS_S8_Mod(mllib_s8 *xz, const mllib_s8 *c,
    mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixAddS_S8_Sat(mllib_s8 *xz, const mllib_s8 *c,
    mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixAddS_S8C_Mod(mllib_s8 *xz, const mllib_s8
    *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixAddS_S8C_Sat(mllib_s8 *xz, const mllib_s8
    *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixAddS_S16_Mod(mllib_s16 *xz, const mllib_s16
    *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixAddS_S16_Sat(mllib_s16 *xz, const mllib_s16
    *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixAddS_S16C_Mod(mllib_s16 *xz, const mllib_s16
    *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixAddS_S16C_Sat(mllib_s16 *xz, const mllib_s16
    *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixAddS_S32_Mod(mllib_s32 *xz, const mllib_s32
    *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixAddS_S32_Sat(mllib_s32 *xz, const mllib_s32
    *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixAddS_S32C_Mod(mllib_s32 *xz, const mllib_s32
    *c, mllib_s32 m, mllib_s32 n);
```

mllib_MatrixAddS_U8_Mod(3MLIB)

```
mllib_status mllib_MatrixAddS_S32C_Sat(mllib_s32 *xz, const mllib_s32
    *c, mllib_s32 m, mllib_s32 n);
```

DESCRIPTION Each of these functions performs an in-place addition of a scalar value to a matrix.

For real data, the following equation is used:

$$xz[i] = c[0] + xz[i]$$

where $i = 0, 1, \dots, (m*n - 1)$.

For complex data, the following equation is used:

$$\begin{aligned}xz[2*i] &= c[0] + xz[2*i] \\xz[2*i + 1] &= c[1] + xz[2*i + 1]\end{aligned}$$

where $i = 0, 1, \dots, (m*n - 1)$.

PARAMETERS Each of the functions takes the following arguments:

xz Pointer to the source and the destination matrix.

c Pointer to the source scalar. When the function is used with complex data types, *c*[0] contains the scalar for the real part, and *c*[1] contains the scalar for the imaginary part.

m Number of rows in the matrices.

n Number of columns in the matrices.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_MatrixAddS_U8_U8_Mod(3MLIB)`, `attributes(5)`

mllib_MatrixAddS_U8_U8_Mod(3MLIB)

NAME | mllib_MatrixAddS_U8_U8_Mod, mllib_MatrixAddS_U8_U8_Sat,
 mllib_MatrixAddS_U8C_U8C_Mod, mllib_MatrixAddS_U8C_U8C_Sat,
 mllib_MatrixAddS_S8_S8_Mod, mllib_MatrixAddS_S8_S8_Sat,
 mllib_MatrixAddS_S8C_S8C_Mod, mllib_MatrixAddS_S8C_S8C_Sat,
 mllib_MatrixAddS_S16_U8_Mod, mllib_MatrixAddS_S16_U8_Sat,
 mllib_MatrixAddS_S16_S8_Mod, mllib_MatrixAddS_S16_S8_Sat,
 mllib_MatrixAddS_S16_S16_Mod, mllib_MatrixAddS_S16_S16_Sat,
 mllib_MatrixAddS_S16C_U8C_Mod, mllib_MatrixAddS_S16C_U8C_Sat,
 mllib_MatrixAddS_S16C_S8C_Mod, mllib_MatrixAddS_S16C_S8C_Sat,
 mllib_MatrixAddS_S16C_S16C_Mod, mllib_MatrixAddS_S16C_S16C_Sat,
 mllib_MatrixAddS_S32_S16_Mod, mllib_MatrixAddS_S32_S16_Sat,
 mllib_MatrixAddS_S32_S32_Mod, mllib_MatrixAddS_S32_S32_Sat,
 mllib_MatrixAddS_S32C_S16C_Mod, mllib_MatrixAddS_S32C_S16C_Sat,
 mllib_MatrixAddS_S32C_S32C_Mod, mllib_MatrixAddS_S32C_S32C_Sat – matrix
 addition to scalar

SYNOPSIS | `cc [flag...] file... -lmllib [library...]
 #include <mllib.h>`

```
mllib_status mllib_MatrixAddS_U8_U8_Mod(mllib_u8 *z, const mllib_u8
    *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixAddS_U8_U8_Sat(mllib_u8 *z, const mllib_u8
    *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixAddS_U8C_U8C_Mod(mllib_u8 *z, const mllib_u8
    *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixAddS_U8C_U8C_Sat(mllib_u8 *z, const mllib_u8
    *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixAddS_S8_S8_Mod(mllib_s8 *z, const mllib_s8
    *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixAddS_S8_S8_Sat(mllib_s8 *z, const mllib_s8
    *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixAddS_S8C_S8C_Mod(mllib_s8 *z, const mllib_s8
    *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixAddS_S8C_S8C_Sat(mllib_s8 *z, const mllib_s8
    *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixAddS_S16_U8_Mod(mllib_s16 *z, const mllib_u8
    *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixAddS_S16_U8_Sat(mllib_s16 *z, const mllib_u8
    *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixAddS_S16_S8_Mod(mllib_s16 *z, const mllib_s8
    *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixAddS_S16_S8_Sat(mllib_s16 *z, const mllib_s8
    *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n);
```

mllib_MatrixAddS_U8_U8_Mod(3MLIB)

```
mllib_status mllib_MatrixAddS_S16_S16_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAddS_S16_S16_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAddS_S16C_U8C_Mod(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAddS_S16C_U8C_Sat(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAddS_S16C_S8C_Mod(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAddS_S16C_S8C_Sat(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAddS_S16C_S16C_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAddS_S16C_S16C_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAddS_S32_S16_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAddS_S32_S16_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAddS_S32_S32_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAddS_S32_S32_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAddS_S32C_S16C_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAddS_S32C_S16C_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAddS_S32C_S32C_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAddS_S32C_S32C_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *c, mllib_s32 m, mllib_s32 n);
```

DESCRIPTION

Each of these functions adds a scalar value to a matrix.

For real data, the following equation is used:

$$z[i] = c[0] + x[i]$$

where $i = 0, 1, \dots, (m*n - 1)$.

For complex data, the following equation is used:

```
z[2*i]      = c[0] + x[2*i]
z[2*i + 1] = c[1] + x[2*i + 1]
```

where $i = 0, 1, \dots, (m*n - 1)$.

PARAMETERS Each of the functions takes the following arguments:

- z* Pointer to the destination matrix.
- x* Pointer to the source matrix.
- c* Pointer to the source scalar. When the function is used with complex data types, *c*[0] contains the scalar for the real part, and *c*[1] contains the scalar for the imaginary part.
- m* Number of rows in the matrices.
- n* Number of columns in the matrices.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_MatrixAddS_U8_U8_Mod(3MLIB)`, `attributes(5)`

mllib_MatrixAdd_U8_Mod(3MLIB)

NAME	mllib_MatrixAdd_U8_Mod, mllib_MatrixAdd_U8_Sat, mllib_MatrixAdd_U8C_Mod, mllib_MatrixAdd_U8C_Sat, mllib_MatrixAdd_S8_Mod, mllib_MatrixAdd_S8_Sat, mllib_MatrixAdd_S8C_Mod, mllib_MatrixAdd_S8C_Sat, mllib_MatrixAdd_S16_Mod, mllib_MatrixAdd_S16_Sat, mllib_MatrixAdd_S16C_Mod, mllib_MatrixAdd_S16C_Sat, mllib_MatrixAdd_S32_Mod, mllib_MatrixAdd_S32_Sat, mllib_MatrixAdd_S32C_Mod, mllib_MatrixAdd_S32C_Sat – matrix addition, in place
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_MatrixAdd_U8_Mod(mllib_u8 *xz, const mllib_u8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixAdd_U8_Sat(mllib_u8 *xz, const mllib_u8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixAdd_U8C_Mod(mllib_u8 *xz, const mllib_u8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixAdd_U8C_Sat(mllib_u8 *xz, const mllib_u8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixAdd_S8_Mod(mllib_s8 *xz, const mllib_s8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixAdd_S8_Sat(mllib_s8 *xz, const mllib_s8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixAdd_S8C_Mod(mllib_s8 *xz, const mllib_s8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixAdd_S8C_Sat(mllib_s8 *xz, const mllib_s8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixAdd_S16_Mod(mllib_s16 *xz, const mllib_s16 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixAdd_S16_Sat(mllib_s16 *xz, const mllib_s16 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixAdd_S16C_Mod(mllib_s16 *xz, const mllib_s16 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixAdd_S16C_Sat(mllib_s16 *xz, const mllib_s16 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixAdd_S32_Mod(mllib_s32 *xz, const mllib_s32 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixAdd_S32_Sat(mllib_s32 *xz, const mllib_s32 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixAdd_S32C_Mod(mllib_s32 *xz, const mllib_s32 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixAdd_S32C_Sat(mllib_s32 *xz, const mllib_s32 *y, mllib_s32 m, mllib_s32 n);</pre>

DESCRIPTION Each of these functions performs an in-place addition of the second source matrix to the first source matrix.

It uses the following equation:

$$xz[i] = xz[i] + y[i]$$

where $i = 0, 1, \dots, (m*n - 1)$ for real data; $i = 0, 1, \dots, (m*n*2 - 1)$ for complex data.

PARAMETERS Each of the functions takes the following arguments:

xz Pointer to the first source and destination matrix.

y Pointer to the second source matrix.

m Number of rows in the matrices.

n Number of columns in the matrices.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_MatrixAdd_U8_U8_Mod\(3MLIB\)](#), [attributes\(5\)](#)

mllib_MatrixAdd_U8_U8_Mod(3MLIB)

NAME	mllib_MatrixAdd_U8_U8_Mod, mllib_MatrixAdd_U8_U8_Sat, mllib_MatrixAdd_U8C_U8C_Mod, mllib_MatrixAdd_U8C_U8C_Sat, mllib_MatrixAdd_S8_S8_Mod, mllib_MatrixAdd_S8_S8_Sat, mllib_MatrixAdd_S8C_S8C_Mod, mllib_MatrixAdd_S8C_S8C_Sat, mllib_MatrixAdd_S16_U8_Mod, mllib_MatrixAdd_S16_U8_Sat, mllib_MatrixAdd_S16_S8_Mod, mllib_MatrixAdd_S16_S8_Sat, mllib_MatrixAdd_S16_S16_Mod, mllib_MatrixAdd_S16_S16_Sat, mllib_MatrixAdd_S16C_U8C_Mod, mllib_MatrixAdd_S16C_U8C_Sat, mllib_MatrixAdd_S16C_S8C_Mod, mllib_MatrixAdd_S16C_S8C_Sat, mllib_MatrixAdd_S16C_S16C_Mod, mllib_MatrixAdd_S16C_S16C_Sat, mllib_MatrixAdd_S32_S16_Mod, mllib_MatrixAdd_S32_S16_Sat, mllib_MatrixAdd_S32_S32_Mod, mllib_MatrixAdd_S32_S32_Sat, mllib_MatrixAdd_S32C_S16C_Mod, mllib_MatrixAdd_S32C_S16C_Sat, mllib_MatrixAdd_S32C_S32C_Mod, mllib_MatrixAdd_S32C_S32C_Sat – matrix addition
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_MatrixAdd_U8_U8_Mod(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixAdd_U8_U8_Sat(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixAdd_U8C_U8C_Mod(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixAdd_U8C_U8C_Sat(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixAdd_S8_S8_Mod(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixAdd_S8_S8_Sat(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixAdd_S8C_S8C_Mod(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixAdd_S8C_S8C_Sat(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixAdd_S16_U8_Mod(mllib_s16 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixAdd_S16_U8_Sat(mllib_s16 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixAdd_S16_S8_Mod(mllib_s16 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixAdd_S16_S8_Sat(mllib_s16 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 n);</pre>

mllib_MatrixAdd_U8_U8_Mod(3MLIB)

```
mllib_status mllib_MatrixAdd_S16_S16_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAdd_S16_S16_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAdd_S16C_U8C_Mod(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAdd_S16C_U8C_Sat(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAdd_S16C_S8C_Mod(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAdd_S16C_S8C_Sat(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAdd_S16C_S16C_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAdd_S16C_S16C_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAdd_S32_S16_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAdd_S32_S16_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAdd_S32_S32_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAdd_S32_S32_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAdd_S32C_S16C_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAdd_S32C_S16C_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAdd_S32C_S32C_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixAdd_S32C_S32C_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 m, mllib_s32 n);
```

DESCRIPTION

Each of these functions adds the first source matrix to the second source matrix and writes the output to the destination matrix.

It uses the following equation:

$$z[i] = x[i] + y[i]$$

where $i = 0, 1, \dots, (m*n - 1)$ for real data; $i = 0, 1, \dots, (m*n*2 - 1)$ for complex data.

`mllib_MatrixAdd_U8_U8_Mod(3MLIB)`

PARAMETERS Each of the functions takes the following arguments:

- z* Pointer to the destination matrix.
- x* Pointer to the first source matrix.
- y* Pointer to the second source matrix.
- m* Number of rows in the matrices.
- n* Number of columns in the matrices.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_MatrixAdd_U8_U8_Mod(3MLIB)`, `attributes(5)`

mllib_MatrixMaximumMag_U8C(3MLIB)

NAME mllib_MatrixMaximumMag_U8C, mllib_MatrixMaximumMag_S8C, mllib_MatrixMaximumMag_S16C, mllib_MatrixMaximumMag_S32C, mllib_MatrixMaximumMag_F32C, mllib_MatrixMaximumMag_D64C – find the first element with the maximum magnitude in a matrix

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_MatrixMaximumMag_U8C(mllib_u8 *max, const mllib_u8
    *x, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixMaximumMag_S8C(mllib_s8 *max, const mllib_s8
    *x, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixMaximumMag_S16C(mllib_s16 *max, const
    mllib_s16 *x, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixMaximumMag_S32C(mllib_s32 *max, const
    mllib_s32 *x, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixMaximumMag_F32C(mllib_f32 *max, const
    mllib_f32 *x, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixMaximumMag_D64C(mllib_d64 *max, const
    mllib_d64 *x, mllib_s32 m, mllib_s32 n);
```

DESCRIPTION Each of these functions finds the first element with the maximum magnitude in a complex matrix, then puts the real and imaginary parts of it into max[0] and max[1], respectively.

PARAMETERS Each of the functions takes the following arguments:

max Pointer to the first element with the maximum magnitude

x Pointer to the first element of the source matrix.

m Number of rows in the source matrix

n Number of columns in the source matrix

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_MatrixMinimumMag_U8C\(3MLIB\)](#), [mllib_VectorMaximumMag_U8C\(3MLIB\)](#), [mllib_VectorMinimumMag_U8C\(3MLIB\)](#), [attributes\(5\)](#)

mllib_MatrixMaximum_U8(3MLIB)

NAME | mllib_MatrixMaximum_U8, mllib_MatrixMaximum_S8, mllib_MatrixMaximum_S16, mllib_MatrixMaximum_S32, mllib_MatrixMaximum_F32, mllib_MatrixMaximum_D64 – find the maximum value in a matrix

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_MatrixMaximum_U8(mllib_u8 *max, const mllib_u8 *x,
mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixMaximum_S8(mllib_s8 *max, const mllib_s8 *x,
mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixMaximum_S16(mllib_s16 *max, const mllib_s16
*x, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixMaximum_S32(mllib_s32 *max, const mllib_s32
*x, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixMaximum_F32(mllib_f32 *max, const mllib_f32
*x, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixMaximum_D64(mllib_d64 *max, const mllib_d64
*x, mllib_s32 m, mllib_s32 n);
```

DESCRIPTION | Each of these functions finds the maximum value of all elements in a matrix. It uses the following equation:

$$\max[0] = \text{MAX}\{ x[i] \quad i = 0, 1, \dots, (m*n - 1) \}$$

PARAMETERS | Each of the functions takes the following arguments:

max | Pointer to the maximum value.

x | Pointer to the first element of the source matrix.

m | Number of rows in the source matrix.

n | Number of columns in the source matrix.

RETURN VALUES | Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_MatrixMinimum_U8(3MLIB), mllib_VectorMaximum_U8(3MLIB), mllib_VectorMinimum_U8(3MLIB), attributes(5)

NAME mllib_MatrixMinimumMag_U8C, mllib_MatrixMinimumMag_S8C, mllib_MatrixMinimumMag_S16C, mllib_MatrixMinimumMag_S32C, mllib_MatrixMinimumMag_F32C, mllib_MatrixMinimumMag_D64C – find the first element with the minimum magnitude in a matrix

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_MatrixMinimumMag_U8C(mllib_u8 *min, const mllib_u8
    *x, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixMinimumMag_S8C(mllib_s8 *min, const mllib_s8
    *x, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixMinimumMag_S16C(mllib_s16 *min, const
    mllib_s16 *x, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixMinimumMag_S32C(mllib_s32 *min, const
    mllib_s32 *x, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixMinimumMag_F32C(mllib_f32 *min, const
    mllib_f32 *x, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixMinimumMag_D64C(mllib_d64 *min, const
    mllib_d64 *x, mllib_s32 m, mllib_s32 n);
```

DESCRIPTION Each of these functions finds the first element with the minimum magnitude in a complex matrix, then puts the real and imaginary parts of it into `min[0]` and `min[1]`, respectively.

PARAMETERS Each of the functions takes the following arguments:

min Pointer to the first element with the minimum magnitude.

x Pointer to the first element of the source matrix.

m Number of rows in the source matrix.

n Number of columns in the source matrix.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_MatrixMaximumMag_U8C\(3MLIB\)](#), [mllib_VectorMaximumMag_U8C\(3MLIB\)](#), [mllib_VectorMinimumMag_U8C\(3MLIB\)](#), [attributes\(5\)](#)

mllib_MatrixMinimum_U8(3MLIB)

NAME | mllib_MatrixMinimum_U8, mllib_MatrixMinimum_S8, mllib_MatrixMinimum_S16, mllib_MatrixMinimum_S32, mllib_MatrixMinimum_F32, mllib_MatrixMinimum_D64 – find the minimum value in a matrix

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_MatrixMinimum_U8(mllib_u8 *min, const mllib_u8 *x,
mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixMinimum_S8(mllib_s8 *min, const mllib_s8 *x,
mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixMinimum_S16(mllib_s16 *min, const mllib_s16
*x, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixMinimum_S32(mllib_s32 *min, const mllib_s32
*x, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixMinimum_F32(mllib_f32 *min, const mllib_f32
*x, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixMinimum_D64(mllib_d64 *min, const mllib_d64
*x, mllib_s32 m, mllib_s32 n);
```

DESCRIPTION | Each of these functions finds the minimum value of all elements in a matrix. It uses the following equation:

$$\min[0] = \text{MIN}\{ x[i] \quad i = 0, 1, \dots, (m*n - 1) \}$$

PARAMETERS | Each of the functions takes the following arguments:

min | Pointer to the minimum value.

x | Pointer to the first element of the source matrix.

m | Number of rows in the source matrix.

n | Number of columns in the source matrix.

RETURN VALUES | Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_MatrixMaximum_U8(3MLIB), mllib_VectorMaximum_U8(3MLIB), mllib_VectorMinimum_U8(3MLIB), attributes(5)

mllib_MatrixMulShift_S16_S16_Mod(3MLIB)

NAME	mllib_MatrixMulShift_S16_S16_Mod, mllib_MatrixMulShift_S16_S16_Sat, mllib_MatrixMulShift_S16C_S16C_Mod, mllib_MatrixMulShift_S16C_S16C_Sat – matrix multiplication plus shifting
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_MatrixMulShift_S16_S16_Mod(mllib_s16 *z, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 l, mllib_s32 n, mllib_s32 shift); mllib_status mllib_MatrixMulShift_S16_S16_Sat(mllib_s16 *z, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 l, mllib_s32 n, mllib_s32 shift); mllib_status mllib_MatrixMulShift_S16C_S16C_Mod(mllib_s16 *z, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 l, mllib_s32 n, mllib_s32 shift); mllib_status mllib_MatrixMulShift_S16C_S16C_Sat(mllib_s16 *z, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 l, mllib_s32 n, mllib_s32 shift);</pre>
DESCRIPTION	<p>Each of these functions performs a multiplication of two matrices and shifts the result.</p> <p>For real data, the following equation is used:</p> $z[i*n + j] = \left\{ \sum_{k=0}^{l-1} (x[i*1 + k] * y[k*n + j]) \right\} * 2^{**(-shift)}$ <p>where $i = 0, 1, \dots, (m - 1)$; $j = 0, 1, \dots, (n - 1)$.</p> <p>For complex data, the following equation is used:</p> $z[2*(i*n + j)] = \left\{ \sum_{k=0}^{l-1} (xR*yR - xI*yI) \right\} * 2^{**(-shift)}$ $z[2*(i*n + j) + 1] = \left\{ \sum_{k=0}^{l-1} (xR*yI + xI*yR) \right\} * 2^{**(-shift)}$ <p>where</p> <pre>xR = x[2*(i*1 + k)] xI = x[2*(i*1 + k) + 1] yR = y[2*(k*n + j)] yI = y[2*(k*n + j) + 1] i = 0, 1, ..., (m - 1) j = 0, 1, ..., (n - 1)</pre>
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <p>z Pointer to the first element of the result matrix, in row major order.</p> <p>x Pointer to the first element of the first matrix, in row major order.</p>

`mllib_MatrixMulShift_S16_S16_Mod(3MLIB)`

y Pointer to the first element of the second matrix, in row major order.

m Number of rows in the first matrix. $m > 0$.

l Number of columns in the first matrix, and the number of rows in the second matrix. $l > 0$.

n Number of columns in the second matrix. $n > 0$.

shift Right shifting factor. $1 \leq \text{shift} \leq 16$.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_MatrixMul_U8_U8_Mod(3MLIB)`, `attributes(5)`

mllib_MatrixMulSShift_U8_Mod(3MLIB)

NAME mllib_MatrixMulSShift_U8_Mod, mllib_MatrixMulSShift_U8_Sat, mllib_MatrixMulSShift_U8C_Mod, mllib_MatrixMulSShift_U8C_Sat, mllib_MatrixMulSShift_S8_Mod, mllib_MatrixMulSShift_S8_Sat, mllib_MatrixMulSShift_S8C_Mod, mllib_MatrixMulSShift_S8C_Sat, mllib_MatrixMulSShift_S16_Mod, mllib_MatrixMulSShift_S16_Sat, mllib_MatrixMulSShift_S16C_Mod, mllib_MatrixMulSShift_S16C_Sat, mllib_MatrixMulSShift_S32_Mod, mllib_MatrixMulSShift_S32_Sat, mllib_MatrixMulSShift_S32C_Mod, mllib_MatrixMulSShift_S32C_Sat – matrix multiplication by scalar plus shifting, in place

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_MatrixMulSShift_U8_Mod(mllib_u8 *xz, const
    mllib_u8 *c, mllib_s32 m, mllib_s32 n, mllib_s32 shift);

mllib_status mllib_MatrixMulSShift_U8_Sat(mllib_u8 *xz, const
    mllib_u8 *c, mllib_s32 m, mllib_s32 n, mllib_s32 shift);

mllib_status mllib_MatrixMulSShift_U8C_Mod(mllib_u8 *xz, const
    mllib_u8 *c, mllib_s32 m, mllib_s32 n, mllib_s32 shift);

mllib_status mllib_MatrixMulSShift_U8C_Sat(mllib_u8 *xz, const
    mllib_u8 *c, mllib_s32 m, mllib_s32 n, mllib_s32 shift);

mllib_status mllib_MatrixMulSShift_S8_Mod(mllib_s8 *xz, const
    mllib_s8 *c, mllib_s32 m, mllib_s32 n, mllib_s32 shift);

mllib_status mllib_MatrixMulSShift_S8_Sat(mllib_s8 *xz, const
    mllib_s8 *c, mllib_s32 m, mllib_s32 n, mllib_s32 shift);

mllib_status mllib_MatrixMulSShift_S8C_Mod(mllib_s8 *xz, const
    mllib_s8 *c, mllib_s32 m, mllib_s32 n, mllib_s32 shift);

mllib_status mllib_MatrixMulSShift_S8C_Sat(mllib_s8 *xz, const
    mllib_s8 *c, mllib_s32 m, mllib_s32 n, mllib_s32 shift);

mllib_status mllib_MatrixMulSShift_S16_Mod(mllib_s16 *xz, const
    mllib_s16 *c, mllib_s32 m, mllib_s32 n, mllib_s32 shift);

mllib_status mllib_MatrixMulSShift_S16_Sat(mllib_s16 *xz, const
    mllib_s16 *c, mllib_s32 m, mllib_s32 n, mllib_s32 shift);

mllib_status mllib_MatrixMulSShift_S16C_Mod(mllib_s16 *xz, const
    mllib_s16 *c, mllib_s32 m, mllib_s32 n, mllib_s32 shift);

mllib_status mllib_MatrixMulSShift_S16C_Sat(mllib_s16 *xz, const
    mllib_s16 *c, mllib_s32 m, mllib_s32 n, mllib_s32 shift);

mllib_status mllib_MatrixMulSShift_S32_Mod(mllib_s32 *xz, const
    mllib_s32 *c, mllib_s32 m, mllib_s32 n, mllib_s32 shift);

mllib_status mllib_MatrixMulSShift_S32_Sat(mllib_s32 *xz, const
    mllib_s32 *c, mllib_s32 m, mllib_s32 n, mllib_s32 shift);
```

mllib_MatrixMulSShift_U8_Mod(3MLIB)

```
mllib_status mllib_MatrixMulSShift_S32C_Mod(mllib_s32 *xz, const
    mllib_s32 *c, mllib_s32 m, mllib_s32 n, mllib_s32 shift);

mllib_status mllib_MatrixMulSShift_S32C_Sat(mllib_s32 *xz, const
    mllib_s32 *c, mllib_s32 m, mllib_s32 n, mllib_s32 shift);
```

DESCRIPTION

Each of these functions performs an in-place multiplication of a matrix with a scalar and shifts the result.

For real data, the following equation is used:

$$xz[i] = c[0]*xz[i]*2^{*(-shift)}$$

where $i = 0, 1, \dots, (m*n - 1)$.

For complex data, the following equation is used:

$$\begin{aligned} tmp &= xz[2*i] \\ xz[2*i] &= (c[0]*tmp - c[1]*xz[2*i + 1])*2^{*(-shift)} \\ xz[2*i + 1] &= (c[1]*tmp + c[0]*xz[2*i + 1])*2^{*(-shift)} \end{aligned}$$

where $i = 0, 1, \dots, (m*n - 1)$.

The ranges of valid shift are:

- 1 ≤ shift ≤ 8 for U8, S8, U8C, S8C types
- 1 ≤ shift ≤ 16 for S16, S16C types
- 1 ≤ shift ≤ 31 for S32, S32C types

PARAMETERS

Each of the functions takes the following arguments:

- xz* Pointer to the source and destination matrix.
- c* Pointer to the source scalar. When the function is used with complex data types, *c*[0] contains the scalar for the real part, and *c*[1] contains the scalar for the imaginary part.
- m* Number of rows in each matrix.
- n* Number of columns in each matrix.
- shift* Right shifting factor.

RETURN VALUES

Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

[mllib_MatrixMulSShift_U8_U8_Mod\(3MLIB\)](#), [attributes\(5\)](#)

mllib_MatrixMulSShift_U8_U8_Mod(3MLIB)

NAME mllib_MatrixMulSShift_U8_U8_Mod, mllib_MatrixMulSShift_U8_U8_Sat, mllib_MatrixMulSShift_U8C_U8C_Mod, mllib_MatrixMulSShift_U8C_U8C_Sat, mllib_MatrixMulSShift_S8_S8_Mod, mllib_MatrixMulSShift_S8_S8_Sat, mllib_MatrixMulSShift_S8C_S8C_Mod, mllib_MatrixMulSShift_S8C_S8C_Sat, mllib_MatrixMulSShift_S16_S16_Mod, mllib_MatrixMulSShift_S16_S16_Sat, mllib_MatrixMulSShift_S16C_S16C_Mod, mllib_MatrixMulSShift_S16C_S16C_Sat, mllib_MatrixMulSShift_S32_S32_Mod, mllib_MatrixMulSShift_S32_S32_Sat, mllib_MatrixMulSShift_S32C_S32C_Mod, mllib_MatrixMulSShift_S32C_S32C_Sat – matrix multiplication by scalar plus shifting

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_MatrixMulSShift_U8_U8_Mod(mllib_u8 *z, const
    mllib_u8 *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n, mllib_s32
    shift);

mllib_status mllib_MatrixMulSShift_U8_U8_Sat(mllib_u8 *z, const
    mllib_u8 *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n, mllib_s32
    shift);

mllib_status mllib_MatrixMulSShift_U8C_U8C_Mod(mllib_u8 *z, const
    mllib_u8 *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n, mllib_s32
    shift);

mllib_status mllib_MatrixMulSShift_U8C_U8C_Sat(mllib_u8 *z, const
    mllib_u8 *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n, mllib_s32
    shift);

mllib_status mllib_MatrixMulSShift_S8_S8_Mod(mllib_s8 *z, const
    mllib_s8 *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n, mllib_s32
    shift);

mllib_status mllib_MatrixMulSShift_S8_S8_Sat(mllib_s8 *z, const
    mllib_s8 *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n, mllib_s32
    shift);

mllib_status mllib_MatrixMulSShift_S8C_S8C_Mod(mllib_s8 *z, const
    mllib_s8 *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n, mllib_s32
    shift);

mllib_status mllib_MatrixMulSShift_S8C_S8C_Sat(mllib_s8 *z, const
    mllib_s8 *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n, mllib_s32
    shift);

mllib_status mllib_MatrixMulSShift_S16_S16_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n,
    mllib_s32 shift);

mllib_status mllib_MatrixMulSShift_S16_S16_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n,
    mllib_s32 shift);
```

mllib_MatrixMulSShift_U8_U8_Mod(3MLIB)

```
mllib_status mllib_MatrixMulSShift_S16C_S16C_Mod(mllib_s16 *z, const
mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n,
mllib_s32 shift);

mllib_status mllib_MatrixMulSShift_S16C_S16C_Sat(mllib_s16 *z, const
mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n,
mllib_s32 shift);

mllib_status mllib_MatrixMulSShift_S32_S32_Mod(mllib_s32 *z, const
mllib_s32 *x, const mllib_s32 *c, mllib_s32 m, mllib_s32 n,
mllib_s32 shift);

mllib_status mllib_MatrixMulSShift_S32_S32_Sat(mllib_s32 *z, const
mllib_s32 *x, const mllib_s32 *c, mllib_s32 m, mllib_s32 n,
mllib_s32 shift);

mllib_status mllib_MatrixMulSShift_S32C_S32C_Mod(mllib_s32 *z, const
mllib_s32 *x, const mllib_s32 *c, mllib_s32 m, mllib_s32 n,
mllib_s32 shift);

mllib_status mllib_MatrixMulSShift_S32C_S32C_Sat(mllib_s32 *z, const
mllib_s32 *x, const mllib_s32 *c, mllib_s32 m, mllib_s32 n,
mllib_s32 shift);
```

DESCRIPTION

Each of these functions performs a multiplication of a matrix with a scalar and shifts the result.

For real data, the following equation is used:

$$z[i] = c[0]*x[i]*2^{*(-shift)}$$

where $i = 0, 1, \dots, (m*n - 1)$.

For complex data, the following equation is used:

$$\begin{aligned} z[2*i] &= (c[0]*x[2*i] - c[1]*x[2*i + 1])*2^{*(-shift)} \\ z[2*i + 1] &= (c[1]*x[2*i] + c[0]*x[2*i + 1])*2^{*(-shift)} \end{aligned}$$

where $i = 0, 1, \dots, (m*n - 1)$.

The ranges of valid shift are:

$1 \leq \text{shift} \leq 8$ for U8, S8, U8C, S8C types
 $1 \leq \text{shift} \leq 16$ for S16, S16C types
 $1 \leq \text{shift} \leq 31$ for S32, S32C types

PARAMETERS

Each of the functions takes the following arguments:

<i>z</i>	Pointer to the destination matrix.
<i>x</i>	Pointer to the source matrix.
<i>c</i>	Pointer to the source scalar. When the function is used with complex data types, <i>c</i> [0] contains the scalar for the real part, and <i>c</i> [1] contains the scalar for the imaginary part.
<i>m</i>	Number of rows in each matrix.

`mllib_MatrixMulSShift_U8_U8_Mod(3MLIB)`

n Number of columns in each matrix.

shift Right shifting factor.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_MatrixMulSShift_U8_Mod(3MLIB)`, `attributes(5)`

mllib_MatrixMulS_U8_Mod(3MLIB)

NAME	mllib_MatrixMulS_U8_Mod, mllib_MatrixMulS_U8_Sat, mllib_MatrixMulS_U8C_Mod, mllib_MatrixMulS_U8C_Sat, mllib_MatrixMulS_S8_Mod, mllib_MatrixMulS_S8_Sat, mllib_MatrixMulS_S8C_Mod, mllib_MatrixMulS_S8C_Sat, mllib_MatrixMulS_S16_Mod, mllib_MatrixMulS_S16_Sat, mllib_MatrixMulS_S16C_Mod, mllib_MatrixMulS_S16C_Sat, mllib_MatrixMulS_S32_Mod, mllib_MatrixMulS_S32_Sat, mllib_MatrixMulS_S32C_Mod, mllib_MatrixMulS_S32C_Sat – matrix multiplication by scalar, in place
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_MatrixMulS_U8_Mod(mllib_u8 *xz, const mllib_u8 *c, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixMulS_U8_Sat(mllib_u8 *xz, const mllib_u8 *c, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixMulS_U8C_Mod(mllib_u8 *xz, const mllib_u8 *c, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixMulS_U8C_Sat(mllib_u8 *xz, const mllib_u8 *c, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixMulS_S8_Mod(mllib_s8 *xz, const mllib_s8 *c, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixMulS_S8_Sat(mllib_s8 *xz, const mllib_s8 *c, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixMulS_S8C_Mod(mllib_s8 *xz, const mllib_s8 *c, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixMulS_S8C_Sat(mllib_s8 *xz, const mllib_s8 *c, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixMulS_S16_Mod(mllib_s16 *xz, const mllib_s16 *c, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixMulS_S16_Sat(mllib_s16 *xz, const mllib_s16 *c, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixMulS_S16C_Mod(mllib_s16 *xz, const mllib_s16 *c, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixMulS_S16C_Sat(mllib_s16 *xz, const mllib_s16 *c, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixMulS_S32_Mod(mllib_s32 *xz, const mllib_s32 *c, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixMulS_S32_Sat(mllib_s32 *xz, const mllib_s32 *c, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixMulS_S32C_Mod(mllib_s32 *xz, const mllib_s32 *c, mllib_s32 m, mllib_s32 n);</pre>

```
mllib_status mllib_MatrixMulS_S32C_Sat(mllib_s32 *xz, const mllib_s32
    *c, mllib_s32 m, mllib_s32 n);
```

DESCRIPTION Each of these functions performs an in-place multiplication of a scalar to a matrix.

For real data, the following equation is used:

$$xz[i] = c[0]*xz[i]$$

where $i = 0, 1, \dots, (m*n - 1)$.

For complex data, the following equation is used:

$$\begin{aligned} tmp &= xz[2*i] \\ xz[2*i] &= c[0]*tmp - c[1]*xz[2*i + 1] \\ xz[2*i + 1] &= c[1]*tmp + c[0]*xz[2*i + 1] \end{aligned}$$

where $i = 0, 1, \dots, (m*n - 1)$.

PARAMETERS Each of the functions takes the following arguments:

- xz* Pointer to the source and destination matrix.
- c* Pointer to the source scalar. When the function is used with complex data types, *c*[0] contains the scalar for the real part, and *c*[1] contains the scalar for the imaginary part.
- m* Number of rows in each matrix.
- n* Number of columns in each matrix.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_MatrixMulS_U8_U8_Mod(3MLIB)`, `attributes(5)`

mllib_MatrixMulS_U8_U8_Mod(3MLIB)

NAME	mllib_MatrixMulS_U8_U8_Mod, mllib_MatrixMulS_U8_U8_Sat, mllib_MatrixMulS_U8C_U8C_Mod, mllib_MatrixMulS_U8C_U8C_Sat, mllib_MatrixMulS_S8_S8_Mod, mllib_MatrixMulS_S8_S8_Sat, mllib_MatrixMulS_S8C_S8C_Mod, mllib_MatrixMulS_S8C_S8C_Sat, mllib_MatrixMulS_S16_U8_Mod, mllib_MatrixMulS_S16_U8_Sat, mllib_MatrixMulS_S16_S8_Mod, mllib_MatrixMulS_S16_S8_Sat, mllib_MatrixMulS_S16_S16_Mod, mllib_MatrixMulS_S16_S16_Sat, mllib_MatrixMulS_S16C_U8C_Mod, mllib_MatrixMulS_S16C_U8C_Sat, mllib_MatrixMulS_S16C_S8C_Mod, mllib_MatrixMulS_S16C_S8C_Sat, mllib_MatrixMulS_S16C_S16C_Mod, mllib_MatrixMulS_S16C_S16C_Sat, mllib_MatrixMulS_S32_S16_Mod, mllib_MatrixMulS_S32_S16_Sat, mllib_MatrixMulS_S32_S32_Mod, mllib_MatrixMulS_S32_S32_Sat, mllib_MatrixMulS_S32C_S16C_Mod, mllib_MatrixMulS_S32C_S16C_Sat, mllib_MatrixMulS_S32C_S32C_Mod, mllib_MatrixMulS_S32C_S32C_Sat – matrix multiplication by scalar
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_MatrixMulS_U8_U8_Mod(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixMulS_U8_U8_Sat(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixMulS_U8C_U8C_Mod(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixMulS_U8C_U8C_Sat(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixMulS_S8_S8_Mod(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixMulS_S8_S8_Sat(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixMulS_S8C_S8C_Mod(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixMulS_S8C_S8C_Sat(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixMulS_S16_U8_Mod(mllib_s16 *z, const mllib_u8 *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixMulS_S16_U8_Sat(mllib_s16 *z, const mllib_u8 *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixMulS_S16_S8_Mod(mllib_s16 *z, const mllib_s8 *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixMulS_S16_S8_Sat(mllib_s16 *z, const mllib_s8 *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n);</pre>

mllib_MatrixMulS_U8_U8_Mod(3MLIB)

```

mllib_status mllib_MatrixMulS_S16_S16_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixMulS_S16_S16_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixMulS_S16C_U8C_Mod(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixMulS_S16C_U8C_Sat(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixMulS_S16C_S8C_Mod(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixMulS_S16C_S8C_Sat(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixMulS_S16C_S16C_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixMulS_S16C_S16C_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixMulS_S32_S16_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixMulS_S32_S16_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixMulS_S32_S32_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixMulS_S32_S32_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixMulS_S32C_S16C_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixMulS_S32C_S16C_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixMulS_S32C_S32C_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixMulS_S32C_S32C_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *c, mllib_s32 m, mllib_s32 n);

```

DESCRIPTION

Each of these functions multiplies a matrix by a scalar.

For real data, the following equation is used:

$$z[i] = c[0]*x[i]$$

where $i = 0, 1, \dots, (m*n - 1)$.

For complex data, the following equation is used:

mllib_MatrixMulS_U8_U8_Mod(3MLIB)

$$\begin{aligned}z[2*i] &= c[0]*x[2*i] - c[1]*x[2*i + 1] \\z[2*i + 1] &= c[1]*x[2*i] + c[0]*x[2*i + 1]\end{aligned}$$

where $i = 0, 1, \dots, (m*n - 1)$.

PARAMETERS Each of the functions takes the following arguments:

z Pointer to the destination matrix.
x Pointer to the source matrix.
c Pointer to the source scalar. When the function is used with complex data types, *c*[0] contains the scalar for the real part, and *c*[1] contains the scalar for the imaginary part.
m Number of rows in each matrix.
n Number of columns in each matrix.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_MatrixMulS_U8_U8_Mod(3MLIB)`, `attributes(5)`

mllib_MatrixMul_U8_U8_Mod(3MLIB)

NAME | mllib_MatrixMul_U8_U8_Mod, mllib_MatrixMul_U8_U8_Sat,
 mllib_MatrixMul_U8C_U8C_Mod, mllib_MatrixMul_U8C_U8C_Sat,
 mllib_MatrixMul_S8_S8_Mod, mllib_MatrixMul_S8_S8_Sat,
 mllib_MatrixMul_S8C_S8C_Mod, mllib_MatrixMul_S8C_S8C_Sat,
 mllib_MatrixMul_S16_U8_Mod, mllib_MatrixMul_S16_U8_Sat,
 mllib_MatrixMul_S16_S8_Mod, mllib_MatrixMul_S16_S8_Sat,
 mllib_MatrixMul_S16_S16_Mod, mllib_MatrixMul_S16_S16_Sat,
 mllib_MatrixMul_S16C_U8C_Mod, mllib_MatrixMul_S16C_U8C_Sat,
 mllib_MatrixMul_S16C_S8C_Mod, mllib_MatrixMul_S16C_S8C_Sat,
 mllib_MatrixMul_S16C_S16C_Mod, mllib_MatrixMul_S16C_S16C_Sat,
 mllib_MatrixMul_S32_S16_Mod, mllib_MatrixMul_S32_S16_Sat,
 mllib_MatrixMul_S32_S32_Mod, mllib_MatrixMul_S32_S32_Sat,
 mllib_MatrixMul_S32C_S16C_Mod, mllib_MatrixMul_S32C_S16C_Sat,
 mllib_MatrixMul_S32C_S32C_Mod, mllib_MatrixMul_S32C_S32C_Sat – matrix
 multiplication

SYNOPSIS | `cc [flag...] file... -lmllib [library...]`

```
#include <mllib.h>

mllib_status mllib_MatrixMul_U8_U8_Mod(mllib_u8 *z, const mllib_u8
    *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 l, mllib_s32 n);

mllib_status mllib_MatrixMul_U8_U8_Sat(mllib_u8 *z, const mllib_u8
    *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 l, mllib_s32 n);

mllib_status mllib_MatrixMul_U8C_U8C_Mod(mllib_u8 *z, const mllib_u8
    *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 l, mllib_s32 n);

mllib_status mllib_MatrixMul_U8C_U8C_Sat(mllib_u8 *z, const mllib_u8
    *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 l, mllib_s32 n);

mllib_status mllib_MatrixMul_S8_S8_Mod(mllib_s8 *z, const mllib_s8
    *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 l, mllib_s32 n);

mllib_status mllib_MatrixMul_S8_S8_Sat(mllib_s8 *z, const mllib_s8
    *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 l, mllib_s32 n);

mllib_status mllib_MatrixMul_S8C_S8C_Mod(mllib_s8 *z, const mllib_s8
    *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 l, mllib_s32 n);

mllib_status mllib_MatrixMul_S8C_S8C_Sat(mllib_s8 *z, const mllib_s8
    *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 l, mllib_s32 n);

mllib_status mllib_MatrixMul_S16_U8_Mod(mllib_s16 *z, const mllib_u8
    *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 l, mllib_s32 n);

mllib_status mllib_MatrixMul_S16_U8_Sat(mllib_s16 *z, const mllib_u8
    *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 l, mllib_s32 n);

mllib_status mllib_MatrixMul_S16_S8_Mod(mllib_s16 *z, const mllib_s8
    *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 l, mllib_s32 n);

mllib_status mllib_MatrixMul_S16_S8_Sat(mllib_s16 *z, const mllib_s8
    *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 l, mllib_s32 n);
```

mllib_MatrixMul_U8_U8_Mod(3MLIB)

```
mllib_status mllib_MatrixMul_S16_S16_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 l,
    mllib_s32 n);

mllib_status mllib_MatrixMul_S16_S16_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 l,
    mllib_s32 n);

mllib_status mllib_MatrixMul_S16C_U8C_Mod(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 l, mllib_s32
    n);

mllib_status mllib_MatrixMul_S16C_U8C_Sat(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 l, mllib_s32
    n);

mllib_status mllib_MatrixMul_S16C_S8C_Mod(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 l, mllib_s32
    n);

mllib_status mllib_MatrixMul_S16C_S8C_Sat(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 l, mllib_s32
    n);

mllib_status mllib_MatrixMul_S16C_S16C_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 l,
    mllib_s32 n);

mllib_status mllib_MatrixMul_S16C_S16C_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 l,
    mllib_s32 n);

mllib_status mllib_MatrixMul_S32_S16_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 l,
    mllib_s32 n);

mllib_status mllib_MatrixMul_S32_S16_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 l,
    mllib_s32 n);

mllib_status mllib_MatrixMul_S32_S32_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 m, mllib_s32 l,
    mllib_s32 n);

mllib_status mllib_MatrixMul_S32_S32_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 m, mllib_s32 l,
    mllib_s32 n);

mllib_status mllib_MatrixMul_S32C_S16C_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 l,
    mllib_s32 n);

mllib_status mllib_MatrixMul_S32C_S16C_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 l,
    mllib_s32 n);
```

```

mllib_status mllib_MatrixMul_S32C_S32C_Mod(mllib_s32 *z, const
mllib_s32 *x, const mllib_s32 *y, mllib_s32 m, mllib_s32 l,
mllib_s32 n);

mllib_status mllib_MatrixMul_S32C_S32C_Sat(mllib_s32 *z, const
mllib_s32 *x, const mllib_s32 *y, mllib_s32 m, mllib_s32 l,
mllib_s32 n);

```

DESCRIPTION Each of these functions performs matrix multiplication of the first matrix to the second matrix or the first complex matrix to the second complex matrix.

For real data, the following equation is used:

$$z[i*n + j] = \sum_{k=0}^{l-1} (x[i*1 + k] * y[k*n + j])$$

where $i = 0, 1, \dots, (m - 1)$; $j = 0, 1, \dots, (n - 1)$.

For complex data, the following equation is used:

$$z[2*(i*n + j)] = \sum_{k=0}^{l-1} (xR*yR - xI*yI)$$

$$z[2*(i*n + j) + 1] = \sum_{k=0}^{l-1} (xR*yI + xI*yR)$$

where

```

xR = x[2*(i*1 + k)]
xI = x[2*(i*1 + k) + 1]
yR = y[2*(k*n + j)]
yI = y[2*(k*n + j) + 1]
i = 0, 1, ..., (m - 1)
j = 0, 1, ..., (n - 1)

```

PARAMETERS Each of the functions takes the following arguments:

<i>z</i>	Pointer to the destination matrix.
<i>x</i>	Pointer to the first source matrix.
<i>y</i>	Pointer to the second source matrix.
<i>m</i>	Number of rows in the first matrix.
<i>l</i>	Number of columns in the first matrix, and number of rows in the second matrix.
<i>n</i>	Number of columns in the second matrix.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

mllib_MatrixMul_U8_U8_Mod(3MLIB)

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_MatrixMulShift_S16_S16_Mod\(3MLIB\)](#), `attributes(5)`

NAME | mllib_MatrixScale_U8_Mod, mllib_MatrixScale_U8_Sat, mllib_MatrixScale_U8C_Mod, mllib_MatrixScale_U8C_Sat, mllib_MatrixScale_S8_Mod, mllib_MatrixScale_S8_Sat, mllib_MatrixScale_S8C_Mod, mllib_MatrixScale_S8C_Sat, mllib_MatrixScale_S16_Mod, mllib_MatrixScale_S16_Sat, mllib_MatrixScale_S16C_Mod, mllib_MatrixScale_S16C_Sat, mllib_MatrixScale_S32_Mod, mllib_MatrixScale_S32_Sat, mllib_MatrixScale_S32C_Mod, mllib_MatrixScale_S32C_Sat – matrix linear scaling, in place

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_MatrixScale_U8_Mod(mllib_u8 *xz, const mllib_u8
    *a, const mllib_u8 *b, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixScale_U8_Sat(mllib_u8 *xz, const mllib_u8
    *a, const mllib_u8 *b, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixScale_U8C_Mod(mllib_u8 *xz, const mllib_u8
    *a, const mllib_u8 *b, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixScale_U8C_Sat(mllib_u8 *xz, const mllib_u8
    *a, const mllib_u8 *b, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixScale_S8_Mod(mllib_s8 *xz, const mllib_s8
    *a, const mllib_s8 *b, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixScale_S8_Sat(mllib_s8 *xz, const mllib_s8
    *a, const mllib_s8 *b, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixScale_S8C_Mod(mllib_s8 *xz, const mllib_s8
    *a, const mllib_s8 *b, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixScale_S8C_Sat(mllib_s8 *xz, const mllib_s8
    *a, const mllib_s8 *b, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixScale_S16_Mod(mllib_s16 *xz, const mllib_s16
    *a, const mllib_s16 *b, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixScale_S16_Sat(mllib_s16 *xz, const mllib_s16
    *a, const mllib_s16 *b, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixScale_S16C_Mod(mllib_s16 *xz, const
    mllib_s16 *a, const mllib_s16 *b, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixScale_S16C_Sat(mllib_s16 *xz, const
    mllib_s16 *a, const mllib_s16 *b, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixScale_S32_Mod(mllib_s32 *xz, const mllib_s32
    *a, const mllib_s32 *b, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixScale_S32_Sat(mllib_s32 *xz, const mllib_s32
    *a, const mllib_s32 *b, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixScale_S32C_Mod(mllib_s32 *xz, const
    mllib_s32 *a, const mllib_s32 *b, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixScale_S32C_Sat(mllib_s32 *xz, const
    mllib_s32 *a, const mllib_s32 *b, mllib_s32 m, mllib_s32 n);
```

mllib_MatrixScale_U8_Mod(3MLIB)

DESCRIPTION Each of these functions performs an in-place multiplication of a matrix by a scalar and then adds an offset.

For real data, the following equation is used:

$$xz[i] = a[0]*xz[i] + b[0]$$

where $i = 0, 1, \dots, (m*n - 1)$.

For complex data, the following equation is used:

$$\begin{aligned} \text{tmp} &= xz[2*i] \\ xz[2*i] &= a[0]*\text{tmp} - a[1]*xz[2*i + 1] + b[0] \\ xz[2*i + 1] &= a[1]*\text{tmp} + a[0]*xz[2*i + 1] + b[1] \end{aligned}$$

where $i = 0, 1, \dots, (m*n - 1)$.

PARAMETERS Each of the functions takes the following arguments:

xz Pointer to the source and destination matrix.

a Pointer to the source scaling factor. When the function is used with complex data types, *a*[0] contains the scalar for the real part, and *a*[1] contains the scalar for the imaginary part.

b Pointer to the source offset. When the function is used with complex data types, *b*[0] contains the offset for the real part, and *b*[1] contains the offset for the imaginary part.

m Number of rows in the matrix.

n Number of columns in the matrix.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_MatrixScale_U8_U8_Mod(3MLIB)`, `attributes(5)`

mllib_MatrixScale_U8_U8_Mod(3MLIB)

NAME | mllib_MatrixScale_U8_U8_Mod, mllib_MatrixScale_U8_U8_Sat,
 mllib_MatrixScale_U8C_U8C_Mod, mllib_MatrixScale_U8C_U8C_Sat,
 mllib_MatrixScale_S8_S8_Mod, mllib_MatrixScale_S8_S8_Sat,
 mllib_MatrixScale_S8C_S8C_Mod, mllib_MatrixScale_S8C_S8C_Sat,
 mllib_MatrixScale_S16_U8_Mod, mllib_MatrixScale_S16_U8_Sat,
 mllib_MatrixScale_S16_S8_Mod, mllib_MatrixScale_S16_S8_Sat,
 mllib_MatrixScale_S16_S16_Mod, mllib_MatrixScale_S16_S16_Sat,
 mllib_MatrixScale_S16C_U8C_Mod, mllib_MatrixScale_S16C_U8C_Sat,
 mllib_MatrixScale_S16C_S8C_Mod, mllib_MatrixScale_S16C_S8C_Sat,
 mllib_MatrixScale_S16C_S16C_Mod, mllib_MatrixScale_S16C_S16C_Sat,
 mllib_MatrixScale_S32_S16_Mod, mllib_MatrixScale_S32_S16_Sat,
 mllib_MatrixScale_S32_S32_Mod, mllib_MatrixScale_S32_S32_Sat,
 mllib_MatrixScale_S32C_S16C_Mod, mllib_MatrixScale_S32C_S16C_Sat,
 mllib_MatrixScale_S32C_S32C_Mod, mllib_MatrixScale_S32C_S32C_Sat – matrix linear scaling

SYNOPSIS | `cc [flag...] file... -lmllib [library...]`
`#include <mllib.h>`

`mllib_status mllib_MatrixScale_U8_U8_Mod(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *a, const mllib_u8 *b, mllib_s32 m, mllib_s32 n);`

`mllib_status mllib_MatrixScale_U8_U8_Sat(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *a, const mllib_u8 *b, mllib_s32 m, mllib_s32 n);`

`mllib_status mllib_MatrixScale_U8C_U8C_Mod(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *a, const mllib_u8 *b, mllib_s32 m, mllib_s32 n);`

`mllib_status mllib_MatrixScale_U8C_U8C_Sat(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *a, const mllib_u8 *b, mllib_s32 m, mllib_s32 n);`

`mllib_status mllib_MatrixScale_S8_S8_Mod(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *a, const mllib_s8 *b, mllib_s32 m, mllib_s32 n);`

`mllib_status mllib_MatrixScale_S8_S8_Sat(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *a, const mllib_s8 *b, mllib_s32 m, mllib_s32 n);`

`mllib_status mllib_MatrixScale_S8C_S8C_Mod(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *a, const mllib_s8 *b, mllib_s32 m, mllib_s32 n);`

`mllib_status mllib_MatrixScale_S8C_S8C_Sat(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *a, const mllib_s8 *b, mllib_s32 m, mllib_s32 n);`

`mllib_status mllib_MatrixScale_S16_U8_Mod(mllib_s16 *z, const mllib_u8 *x, const mllib_u8 *a, const mllib_u8 *b, mllib_s32 m, mllib_s32 n);`

mllib_MatrixScale_U8_U8_Mod(3MLIB)

```
mllib_status mllib_MatrixScale_S16_U8_Sat(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *a, const mllib_u8 *b, mllib_s32 m,
    mllib_s32 n);

mllib_status mllib_MatrixScale_S16_S8_Mod(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *a, const mllib_s8 *b, mllib_s32 m,
    mllib_s32 n);

mllib_status mllib_MatrixScale_S16_S8_Sat(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *a, const mllib_s8 *b, mllib_s32 m,
    mllib_s32 n);

mllib_status mllib_MatrixScale_S16_S16_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *a, const mllib_s16 *b, mllib_s32 m,
    mllib_s32 n);

mllib_status mllib_MatrixScale_S16_S16_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *a, const mllib_s16 *b, mllib_s32 m,
    mllib_s32 n);

mllib_status mllib_MatrixScale_S16C_U8C_Mod(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *a, const mllib_u8 *b, mllib_s32 m,
    mllib_s32 n);

mllib_status mllib_MatrixScale_S16C_U8C_Sat(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *a, const mllib_u8 *b, mllib_s32 m,
    mllib_s32 n);

mllib_status mllib_MatrixScale_S16C_S8C_Mod(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *a, const mllib_s8 *b, mllib_s32 m,
    mllib_s32 n);

mllib_status mllib_MatrixScale_S16C_S8C_Sat(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *a, const mllib_s8 *b, mllib_s32 m,
    mllib_s32 n);

mllib_status mllib_MatrixScale_S16C_S16C_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *a, const mllib_s16 *b, mllib_s32 m,
    mllib_s32 n);

mllib_status mllib_MatrixScale_S16C_S16C_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *a, const mllib_s16 *b, mllib_s32 m,
    mllib_s32 n);

mllib_status mllib_MatrixScale_S32_S16_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *a, const mllib_s16 *b, mllib_s32 m,
    mllib_s32 n);

mllib_status mllib_MatrixScale_S32_S16_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *a, const mllib_s16 *b, mllib_s32 m,
    mllib_s32 n);

mllib_status mllib_MatrixScale_S32_S32_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *a, const mllib_s32 *b, mllib_s32 m,
    mllib_s32 n);
```

mllib_MatrixScale_U8_U8_Mod(3MLIB)

```
mllib_status mllib_MatrixScale_S32_S32_Sat(mllib_s32 *z, const
mllib_s32 *x, const mllib_s32 *a, const mllib_s32 *b, mllib_s32 m,
mllib_s32 n);

mllib_status mllib_MatrixScale_S32C_S16C_Mod(mllib_s32 *z, const
mllib_s16 *x, const mllib_s16 *a, const mllib_s16 *b, mllib_s32 m,
mllib_s32 n);

mllib_status mllib_MatrixScale_S32C_S16C_Sat(mllib_s32 *z, const
mllib_s16 *x, const mllib_s16 *a, const mllib_s16 *b, mllib_s32 m,
mllib_s32 n);

mllib_status mllib_MatrixScale_S32C_S32C_Mod(mllib_s32 *z, const
mllib_s32 *x, const mllib_s32 *a, const mllib_s32 *b, mllib_s32 m,
mllib_s32 n);

mllib_status mllib_MatrixScale_S32C_S32C_Sat(mllib_s32 *z, const
mllib_s32 *x, const mllib_s32 *a, const mllib_s32 *b, mllib_s32 m,
mllib_s32 n);
```

DESCRIPTION Each of these functions multiplies a matrix by a scalar and then adds an offset.

For real data, the following equation is used:

$$z[i] = a[0]*x[i] + b[0]$$

where $i = 0, 1, \dots, (m*n - 1)$.

For complex data, the following equation is used:

$$\begin{aligned} z[2*i] &= a[0]*x[2*i] - a[1]*x[2*i + 1] + b[0] \\ z[2*i + 1] &= a[1]*x[2*i] + a[0]*x[2*i + 1] + b[1] \end{aligned}$$

where $i = 0, 1, \dots, (m*n - 1)$.

PARAMETERS Each of the functions takes the following arguments:

z Pointer to the destination matrix.

x Pointer to the source matrix.

a Pointer to the source scaling factor. When the function is used with complex data types, *a*[0] contains the scalar for the real part, and *a*[1] contains the scalar for the imaginary part.

b Pointer to the source offset. When the function is used with complex data types, *b*[0] contains the offset for the real part, and *b*[1] contains the offset for the imaginary part.

m Number of rows in each matrix.

n Number of columns in each matrix.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

mllib_MatrixScale_U8_U8_Mod(3MLIB)

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_MatrixScale_U8_Mod\(3MLIB\)](#), `attributes(5)`

mllib_MatrixSubS_U8_Mod(3MLIB)

NAME mllib_MatrixSubS_U8_Mod, mllib_MatrixSubS_U8_Sat, mllib_MatrixSubS_U8C_Mod, mllib_MatrixSubS_U8C_Sat, mllib_MatrixSubS_S8_Mod, mllib_MatrixSubS_S8_Sat, mllib_MatrixSubS_S8C_Mod, mllib_MatrixSubS_S8C_Sat, mllib_MatrixSubS_S16_Mod, mllib_MatrixSubS_S16_Sat, mllib_MatrixSubS_S16C_Mod, mllib_MatrixSubS_S16C_Sat, mllib_MatrixSubS_S32_Mod, mllib_MatrixSubS_S32_Sat, mllib_MatrixSubS_S32C_Mod, mllib_MatrixSubS_S32C_Sat – matrix subtraction from scalar, in place

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_MatrixSubS_U8_Mod(mllib_u8 *xz, const mllib_u8 *c,
    mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixSubS_U8_Sat(mllib_u8 *xz, const mllib_u8 *c,
    mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixSubS_U8C_Mod(mllib_u8 *xz, const mllib_u8
    *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixSubS_U8C_Sat(mllib_u8 *xz, const mllib_u8
    *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixSubS_S8_Mod(mllib_s8 *xz, const mllib_s8 *c,
    mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixSubS_S8_Sat(mllib_s8 *xz, const mllib_s8 *c,
    mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixSubS_S8C_Mod(mllib_s8 *xz, const mllib_s8
    *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixSubS_S8C_Sat(mllib_s8 *xz, const mllib_s8
    *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixSubS_S16_Mod(mllib_s16 *xz, const mllib_s16
    *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixSubS_S16_Sat(mllib_s16 *xz, const mllib_s16
    *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixSubS_S16C_Mod(mllib_s16 *xz, const mllib_s16
    *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixSubS_S16C_Sat(mllib_s16 *xz, const mllib_s16
    *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixSubS_S32_Mod(mllib_s32 *xz, const mllib_s32
    *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixSubS_S32_Sat(mllib_s32 *xz, const mllib_s32
    *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixSubS_S32C_Mod(mllib_s32 *xz, const mllib_s32
    *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixSubS_S32C_Sat(mllib_s32 *xz, const mllib_s32
    *c, mllib_s32 m, mllib_s32 n);
```

mllib_MatrixSubS_U8_Mod(3MLIB)

DESCRIPTION Each of these functions performs an in-place subtraction of a matrix from a scalar.

For real data, the following equation is used:

$$xz[i] = c[0] - xz[i]$$

where $i = 0, 1, \dots, (m*n - 1)$.

For complex data, the following equation is used:

$$\begin{aligned}xz[2*i] &= c[0] - xz[2*i] \\xz[2*i + 1] &= c[1] - xz[2*i + 1]\end{aligned}$$

where $i = 0, 1, \dots, (m*n - 1)$.

PARAMETERS Each of the functions takes the following arguments:

xz Pointer to the source and destination matrix.
c Pointer to the source scalar. When the function is used with complex data types, *c*[0] contains the scalar for the real part, and *c*[1] contains the scalar for the imaginary part.
m Number of rows in the matrices.
n Number of columns in the matrices.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_MatrixSubS_U8_U8_Mod(3MLIB)`, `attributes(5)`

mllib_MatrixSubS_U8_U8_Mod(3MLIB)

NAME | mllib_MatrixSubS_U8_U8_Mod, mllib_MatrixSubS_U8_U8_Sat,
 mllib_MatrixSubS_U8C_U8C_Mod, mllib_MatrixSubS_U8C_U8C_Sat,
 mllib_MatrixSubS_S8_S8_Mod, mllib_MatrixSubS_S8_S8_Sat,
 mllib_MatrixSubS_S8C_S8C_Mod, mllib_MatrixSubS_S8C_S8C_Sat,
 mllib_MatrixSubS_S16_U8_Mod, mllib_MatrixSubS_S16_U8_Sat,
 mllib_MatrixSubS_S16_S8_Mod, mllib_MatrixSubS_S16_S8_Sat,
 mllib_MatrixSubS_S16_S16_Mod, mllib_MatrixSubS_S16_S16_Sat,
 mllib_MatrixSubS_S16C_U8C_Mod, mllib_MatrixSubS_S16C_U8C_Sat,
 mllib_MatrixSubS_S16C_S8C_Mod, mllib_MatrixSubS_S16C_S8C_Sat,
 mllib_MatrixSubS_S16C_S16C_Mod, mllib_MatrixSubS_S16C_S16C_Sat,
 mllib_MatrixSubS_S32_S16_Mod, mllib_MatrixSubS_S32_S16_Sat,
 mllib_MatrixSubS_S32_S32_Mod, mllib_MatrixSubS_S32_S32_Sat,
 mllib_MatrixSubS_S32C_S16C_Mod, mllib_MatrixSubS_S32C_S16C_Sat,
 mllib_MatrixSubS_S32C_S32C_Mod, mllib_MatrixSubS_S32C_S32C_Sat – matrix
 subtraction from scalar

SYNOPSIS | `cc [flag...] file... -lmllib [library...]
 #include <mllib.h>`

```
mllib_status mllib_MatrixSubS_U8_U8_Mod(mllib_u8 *z, const mllib_u8
    *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixSubS_U8_U8_Sat(mllib_u8 *z, const mllib_u8
    *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixSubS_U8C_U8C_Mod(mllib_u8 *z, const mllib_u8
    *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixSubS_U8C_U8C_Sat(mllib_u8 *z, const mllib_u8
    *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixSubS_S8_S8_Mod(mllib_s8 *z, const mllib_s8
    *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixSubS_S8_S8_Sat(mllib_s8 *z, const mllib_s8
    *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixSubS_S8C_S8C_Mod(mllib_s8 *z, const mllib_s8
    *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixSubS_S8C_S8C_Sat(mllib_s8 *z, const mllib_s8
    *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixSubS_S16_U8_Mod(mllib_s16 *z, const mllib_u8
    *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixSubS_S16_U8_Sat(mllib_s16 *z, const mllib_u8
    *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixSubS_S16_S8_Mod(mllib_s16 *z, const mllib_s8
    *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n);

mllib_status mllib_MatrixSubS_S16_S8_Sat(mllib_s16 *z, const mllib_s8
    *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n);
```

mllib_MatrixSubS_U8_U8_Mod(3MLIB)

```
mllib_status mllib_MatrixSubS_S16_S16_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSubS_S16_S16_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSubS_S16C_U8C_Mod(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSubS_S16C_U8C_Sat(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSubS_S16C_S8C_Mod(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSubS_S16C_S8C_Sat(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSubS_S16C_S16C_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSubS_S16C_S16C_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSubS_S32_S16_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSubS_S32_S16_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSubS_S32_S32_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSubS_S32_S32_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSubS_S32C_S16C_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSubS_S32C_S16C_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSubS_S32C_S32C_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *c, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSubS_S32C_S32C_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *c, mllib_s32 m, mllib_s32 n);
```

DESCRIPTION

Each of these functions subtracts a matrix from a scalar.

For real data, the following equation is used:

$$z[i] = c[0] - x[i]$$

where $i = 0, 1, \dots, (m*n - 1)$.

For complex data, the following equation is used:


```
z[2*i]      = c[0] - x[2*i]
z[2*i + 1] = c[1] - x[2*i + 1]
```

where $i = 0, 1, \dots, (m*n - 1)$.

PARAMETERS Each of the functions takes the following arguments:

- z* Pointer to the destination matrix.
- x* Pointer to the source matrix.
- c* Pointer to the source scalar. When the function is used with complex data types, *c*[0] contains the scalar for the real part, and *c*[1] contains the scalar for the imaginary part.
- m* Number of rows in the matrices.
- n* Number of columns in the matrices.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_MatrixSubS_U8_U8_Mod(3MLIB)`, `attributes(5)`

mllib_MatrixSub_U8_Mod(3MLIB)

NAME	mllib_MatrixSub_U8_Mod, mllib_MatrixSub_U8_Sat, mllib_MatrixSub_U8C_Mod, mllib_MatrixSub_U8C_Sat, mllib_MatrixSub_S8_Mod, mllib_MatrixSub_S8_Sat, mllib_MatrixSub_S8C_Mod, mllib_MatrixSub_S8C_Sat, mllib_MatrixSub_S16_Mod, mllib_MatrixSub_S16_Sat, mllib_MatrixSub_S16C_Mod, mllib_MatrixSub_S16C_Sat, mllib_MatrixSub_S32_Mod, mllib_MatrixSub_S32_Sat, mllib_MatrixSub_S32C_Mod, mllib_MatrixSub_S32C_Sat – matrix subtraction, in place
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_MatrixSub_U8_Mod(mllib_u8 *xz, const mllib_u8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixSub_U8_Sat(mllib_u8 *xz, const mllib_u8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixSub_U8C_Mod(mllib_u8 *xz, const mllib_u8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixSub_U8C_Sat(mllib_u8 *xz, const mllib_u8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixSub_S8_Mod(mllib_s8 *xz, const mllib_s8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixSub_S8_Sat(mllib_s8 *xz, const mllib_s8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixSub_S8C_Mod(mllib_s8 *xz, const mllib_s8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixSub_S8C_Sat(mllib_s8 *xz, const mllib_s8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixSub_S16_Mod(mllib_s16 *xz, const mllib_s16 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixSub_S16_Sat(mllib_s16 *xz, const mllib_s16 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixSub_S16C_Mod(mllib_s16 *xz, const mllib_s16 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixSub_S16C_Sat(mllib_s16 *xz, const mllib_s16 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixSub_S32_Mod(mllib_s32 *xz, const mllib_s32 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixSub_S32_Sat(mllib_s32 *xz, const mllib_s32 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixSub_S32C_Mod(mllib_s32 *xz, const mllib_s32 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixSub_S32C_Sat(mllib_s32 *xz, const mllib_s32 *y, mllib_s32 m, mllib_s32 n);</pre>

DESCRIPTION Each of these functions performs an in-place subtraction of the second matrix from the first matrix.

It uses the following equation:

$$xz[i] = xz[i] - y[i]$$

where $i = 0, 1, \dots, (m*n - 1)$ for real data; $i = 0, 1, \dots, (m*n*2 - 1)$ for complex data.

PARAMETERS Each of the functions takes the following arguments:

xz Pointer to the first source and destination matrix.

y Pointer to the second source matrix.

m Number of rows in the matrices.

n Number of columns in the matrices.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_MatrixSub_U8_U8_Mod\(3MLIB\)](#), `attributes(5)`

mllib_MatrixSub_U8_U8_Mod(3MLIB)

NAME	mllib_MatrixSub_U8_U8_Mod, mllib_MatrixSub_U8_U8_Sat, mllib_MatrixSub_U8C_U8C_Mod, mllib_MatrixSub_U8C_U8C_Sat, mllib_MatrixSub_S8_S8_Mod, mllib_MatrixSub_S8_S8_Sat, mllib_MatrixSub_S8C_S8C_Mod, mllib_MatrixSub_S8C_S8C_Sat, mllib_MatrixSub_S16_U8_Mod, mllib_MatrixSub_S16_U8_Sat, mllib_MatrixSub_S16_S8_Mod, mllib_MatrixSub_S16_S8_Sat, mllib_MatrixSub_S16_S16_Mod, mllib_MatrixSub_S16_S16_Sat, mllib_MatrixSub_S16C_U8C_Mod, mllib_MatrixSub_S16C_U8C_Sat, mllib_MatrixSub_S16C_S8C_Mod, mllib_MatrixSub_S16C_S8C_Sat, mllib_MatrixSub_S16C_S16C_Mod, mllib_MatrixSub_S16C_S16C_Sat, mllib_MatrixSub_S32_S16_Mod, mllib_MatrixSub_S32_S16_Sat, mllib_MatrixSub_S32_S32_Mod, mllib_MatrixSub_S32_S32_Sat, mllib_MatrixSub_S32C_S16C_Mod, mllib_MatrixSub_S32C_S16C_Sat, mllib_MatrixSub_S32C_S32C_Mod, mllib_MatrixSub_S32C_S32C_Sat – matrix subtraction
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_MatrixSub_U8_U8_Mod(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixSub_U8_U8_Sat(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixSub_U8C_U8C_Mod(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixSub_U8C_U8C_Sat(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixSub_S8_S8_Mod(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixSub_S8_S8_Sat(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixSub_S8C_S8C_Mod(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixSub_S8C_S8C_Sat(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixSub_S16_U8_Mod(mllib_s16 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixSub_S16_U8_Sat(mllib_s16 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixSub_S16_S8_Mod(mllib_s16 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixSub_S16_S8_Sat(mllib_s16 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 n);</pre>

mllib_MatrixSub_U8_U8_Mod(3MLIB)

```

mllib_status mllib_MatrixSub_S16_S16_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSub_S16_S16_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSub_S16C_U8C_Mod(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSub_S16C_U8C_Sat(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSub_S16C_S8C_Mod(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSub_S16C_S8C_Sat(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSub_S16C_S16C_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSub_S16C_S16C_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSub_S32_S16_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSub_S32_S16_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSub_S32_S32_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSub_S32_S32_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSub_S32C_S16C_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSub_S32C_S16C_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSub_S32C_S32C_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_MatrixSub_S32C_S32C_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 m, mllib_s32 n);

```

DESCRIPTION

Each of these functions subtracts the second matrix from the first matrix.

It uses the following equation:

$$z[i] = x[i] - y[i]$$

where $i = 0, 1, \dots, (m*n - 1)$ for real data; $i = 0, 1, \dots, (m*n*2 - 1)$ for complex data.

`mllib_MatrixSub_U8_U8_Mod(3MLIB)`

PARAMETERS Each of the functions takes the following arguments:

- z* Pointer to the destination matrix.
- x* Pointer to the first source matrix.
- y* Pointer to the second source matrix.
- m* Number of rows in the matrices.
- n* Number of columns in the matrices.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_MatrixSub_U8_Mod(3MLIB)`, `attributes(5)`

NAME	mllib_MatrixTranspose_U8, mllib_MatrixTranspose_U8C, mllib_MatrixTranspose_S8, mllib_MatrixTranspose_S8C, mllib_MatrixTranspose_S16, mllib_MatrixTranspose_S16C, mllib_MatrixTranspose_S32, mllib_MatrixTranspose_S32C – matrix transpose, in place
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_MatrixTranspose_U8(mllib_u8 *xz, mllib_s32 mn); mllib_status mllib_MatrixTranspose_U8C(mllib_u8 *xz, mllib_s32 mn); mllib_status mllib_MatrixTranspose_S8(mllib_s8 *xz, mllib_s32 mn); mllib_status mllib_MatrixTranspose_S8C(mllib_s8 *xz, mllib_s32 mn); mllib_status mllib_MatrixTranspose_S16(mllib_s16 *xz, mllib_s32 mn); mllib_status mllib_MatrixTranspose_S16C(mllib_s16 *xz, mllib_s32 mn); mllib_status mllib_MatrixTranspose_S32(mllib_s32 *xz, mllib_s32 mn); mllib_status mllib_MatrixTranspose_S32C(mllib_s32 *xz, mllib_s32 mn);</pre>
DESCRIPTION	<p>Each of these functions performs an in-place transpose of a square matrix.</p> <p>For real data, the following pseudo code applies:</p> <pre>for (i = 1; i < mn; i++) { for (j = 0; j < i; i++) { tmp = xz[i*mn + j]; xz[i*mn + j] = xz[j*mn + i]; xz[j*mn + i] = tmp; } }</pre> <p>For complex data, the following pseudo code applies:</p> <pre>for (i = 1; i < mn; i++) { for (j = 0; j < i; i++) { tmp0 = xz[2*(i*mn + j)]; tmp1 = xz[2*(i*mn + j) + 1]; xz[2*(i*mn + j)] = xz[2*(j*mn + i)]; xz[2*(i*mn + j) + 1] = xz[2*(j*mn + i) + 1]; xz[2*(j*mn + i)] = tmp0; xz[2*(j*mn + i) + 1] = tmp1; } }</pre>
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <p><i>xz</i> Pointer to the source and destination matrix.</p> <p><i>mn</i> Number of rows and columns in the matrix.</p>
RETURN VALUES	Each of the functions returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .

mllib_MatrixTranspose_U8(3MLIB)

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_MatrixTranspose_U8\(3MLIB\)](#), `attributes(5)`

mllib_MatrixTranspose_U8_U8(3MLIB)

NAME	mllib_MatrixTranspose_U8_U8, mllib_MatrixTranspose_U8C_U8C, mllib_MatrixTranspose_S8_S8, mllib_MatrixTranspose_S8C_S8C, mllib_MatrixTranspose_S16_S16, mllib_MatrixTranspose_S16C_S16C, mllib_MatrixTranspose_S32_S32, mllib_MatrixTranspose_S32C_S32C – matrix transpose
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_MatrixTranspose_U8_U8(mllib_u8 *z, const mllib_u8 *x, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixTranspose_U8C_U8C(mllib_u8 *z, const mllib_u8 *x, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixTranspose_S8_S8(mllib_s8 *z, const mllib_s8 *x, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixTranspose_S8C_S8C(mllib_s8 *z, const mllib_s8 *x, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixTranspose_S16_S16(mllib_s16 *z, const mllib_s16 *x, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixTranspose_S16C_S16C(mllib_s16 *z, const mllib_s16 *x, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixTranspose_S32_S32(mllib_s32 *z, const mllib_s32 *x, mllib_s32 m, mllib_s32 n); mllib_status mllib_MatrixTranspose_S32C_S32C(mllib_s32 *z, const mllib_s32 *x, mllib_s32 m, mllib_s32 n);</pre>
DESCRIPTION	<p>Each of these functions computes the transpose of the input matrix.</p> <p>For real data, the following equation is used:</p> $z[j*m + i] = x[i*n + j]$ <p>where $i = 0, 1, \dots, (m - 1)$; $j = 0, 1, \dots, (n - 1)$.</p> <p>For complex data, the following equation is used:</p> $z[2*(j*m + i)] = x[2*(i*n + j)]$ $z[2*(j*m + i) + 1] = x[2*(i*n + j) + 1]$ <p>where $i = 0, 1, \dots, (m - 1)$; $j = 0, 1, \dots, (n - 1)$.</p>
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <p><i>z</i> Pointer to the destination matrix. The output data type must be the same as the input data type.</p> <p><i>x</i> Pointer to the source matrix.</p> <p><i>m</i> Number of rows in the source matrix.</p> <p><i>n</i> Number of columns in the source matrix.</p>

`mllib_MatrixTranspose_U8_U8(3MLIB)`

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_MatrixTranspose_U8(3MLIB)`, `attributes(5)`

NAME mllib_MatrixUnit_U8, mllib_MatrixUnit_U8C, mllib_MatrixUnit_S8, mllib_MatrixUnit_S8C, mllib_MatrixUnit_S16, mllib_MatrixUnit_S16C, mllib_MatrixUnit_S32, mllib_MatrixUnit_S32C – Unit matrix generation

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_MatrixUnit_U8(mllib_u8 *z, mllib_s32 n);
mllib_status mllib_MatrixUnit_U8C(mllib_u8 *z, mllib_s32 n);
mllib_status mllib_MatrixUnit_S8(mllib_s8 *z, mllib_s32 n);
mllib_status mllib_MatrixUnit_S8C(mllib_s8 *z, mllib_s32 n);
mllib_status mllib_MatrixUnit_S16(mllib_s16 *z, mllib_s32 n);
mllib_status mllib_MatrixUnit_S16C(mllib_s16 *z, mllib_s32 n);
mllib_status mllib_MatrixUnit_S32(mllib_s32 *z, mllib_s32 n);
mllib_status mllib_MatrixUnit_S32C(mllib_s32 *z, mllib_s32 n);
```

DESCRIPTION Each of these functions sets the values for a unit matrix.

For real data, the following equation is used:

$$z[i*n + j] = 1 \quad \text{if } i == j$$

$$z[i*n + j] = 0 \quad \text{if } i \neq j$$

where $i = 0, 1, \dots, (n - 1)$; $j = 0, 1, \dots, (n - 1)$.

For complex data, the following equation is used:

$$z[2*(i*n + j)] = 1 \quad \text{if } i == j$$

$$z[2*(i*n + j)] = 0 \quad \text{if } i \neq j$$

$$z[2*(i*n + j) + 1] = 0$$

where $i = 0, 1, \dots, (n - 1)$; $j = 0, 1, \dots, (n - 1)$.

PARAMETERS Each of the functions takes the following arguments:

z Pointer to the destination matrix.

n Number of rows and columns in the matrix.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

mllib_MatrixUnit_U8(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	MT-Safe

SEE ALSO [attributes\(5\)](#)

NAME mlib_memcpy – copy a block of bytes

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

void *mlib_memcpy(void *dst, const void *src, size_t n);
```

DESCRIPTION The mlib_memcpy() function copies *n* bytes from memory area *src* to *dst*. It returns *dst*. The memory areas may not overlap. Use mlib_memmove() if the memory areas do overlap.

This function is a wrapper of the standard C function memcpy().

PARAMETERS The function takes the following arguments:

dst Pointer to the destination.

src Pointer to the source.

n Number of bytes to be copied.

RETURN VALUES The function returns a pointer to the destination.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mlib_memmove(3MLIB), mlib_memset(3MLIB), memory(3C), attributes(5)

mllib_memmove(3MLIB)

NAME | mllib_memmove – copy a block of bytes

SYNOPSIS | cc [*flag...*] *file...* -lmllib [*library...*]
| #include <mllib.h>

| void ***mllib_memmove**(void **dst*, const void **src*, size_t *n*);

DESCRIPTION | The mllib_memmove() function copies *n* bytes from memory area *src* to *dst*. Copying between objects that overlap will take place correctly. It returns *dst*.

| This function is a wrapper of the standard C function memmove().

PARAMETERS | The function takes the following arguments:

| *dst* | Pointer to the destination.

| *src* | Pointer to the source.

| *n* | Number of bytes to be copied.

RETURN VALUES | The function returns a pointer to the destination.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_memcpy(3MLIB), mllib_memset(3MLIB), memory(3C), attributes(5)

NAME	mlib_memset – set a block of bytes						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmlib [<i>library...</i>] #include <mlib.h> void *mlib_memset(void *<i>s</i>, mlib_s32 <i>c</i>, size_t <i>n</i>);</pre>						
DESCRIPTION	<p>The <code>mlib_memset()</code> function sets the first <i>n</i> bytes in memory area <i>s</i> to the value of <i>c</i> (converted to an unsigned char). It returns <i>s</i>.</p> <p>This function is a wrapper of the standard C function <code>memset()</code>.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>s</i> Pointer to the destination.</p> <p><i>c</i> Value to set.</p> <p><i>n</i> Number of bytes to be set.</p>						
RETURN VALUES	The function returns a pointer to the destination.						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mlib_memcpy(3MLIB)</code> , <code>mlib_memmove(3MLIB)</code> , <code>memory(3C)</code> , <code>attributes(5)</code>						

mllib_realloc(3MLIB)

NAME | mllib_realloc – reallocate a block of bytes

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
void *mllib_realloc(void *ptr, size_t size);
```

DESCRIPTION | The mllib_realloc() function changes the size of the block pointed to by *ptr* to *size* bytes and returns a pointer to the (possibly moved) block.

This function is a wrapper of the standard C function realloc().

PARAMETERS | The function takes the following arguments:

size New size of the block in bytes.

ptr Pointer to a block.

RETURN VALUES | The function returns a pointer to the reallocated block if successful. Otherwise it returns a null pointer.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_free\(3MLIB\)](#), [mllib_malloc\(3MLIB\)](#), [malloc\(3C\)](#), [attributes\(5\)](#)

mlib_SignalADPCM2Bits2Linear(3MLIB)

NAME mlib_SignalADPCM2Bits2Linear – adaptive differential pulse code modulation (ADPCM)

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_SignalADPCM2Bits2Linear(mlib_s16 *pcm, const
    mlib_u8 *adpcm, void *state, mlib_s32 n);
```

DESCRIPTION The `mlib_SignalADPCM2Bits2Linear()` function performs adaptive differential pulse code modulation (ADPCM) in compliance with the ITU (former CCITT) G.721, G.723, and G.726 specifications. It converts data from G.723 or G.726 16kbps 2-bit ADPCM to 16-bit linear PCM format.

PARAMETERS The function takes the following arguments:

pcm Linear PCM sample array.

adpcm ADPCM code array.

state Internal structure of the codec.

n Number of samples in the source array.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mlib_SignalADPCM3Bits2Linear(3MLIB)`,
`mlib_SignalADPCM4Bits2Linear(3MLIB)`,
`mlib_SignalADPCM5Bits2Linear(3MLIB)`, `mlib_SignalADPCMFree(3MLIB)`,
`mlib_SignalADPCMInit(3MLIB)`, `mlib_SignalLinear2ADPCM2Bits(3MLIB)`,
`mlib_SignalLinear2ADPCM3Bits(3MLIB)`,
`mlib_SignalLinear2ADPCM4Bits(3MLIB)`,
`mlib_SignalLinear2ADPCM5Bits(3MLIB)`, `attributes(5)`

mllib_SignalADPCM3Bits2Linear(3MLIB)

NAME	mllib_SignalADPCM3Bits2Linear – adaptive differential pulse code modulation (ADPCM)						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalADPCM3Bits2Linear(mllib_s16 *<i>pcm</i>, const mllib_u8 *<i>adpcm</i>, void *<i>state</i>, mllib_s32 <i>n</i>);</pre>						
DESCRIPTION	The <code>mllib_SignalADPCM3Bits2Linear()</code> function performs adaptive differential pulse code modulation (ADPCM) in compliance with the ITU (former CCITT) G.721, G.723, and G.726 specifications. It converts data from G.723 or G.726 24kbps 3-bit ADPCM to 16-bit linear PCM format.						
PARAMETERS	The function takes the following arguments: <i>pcm</i> Linear PCM sample array. <i>adpcm</i> ADPCM code array. <i>state</i> Internal structure of the codec. <i>n</i> Number of samples in the source array.						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_SignalADPCM2Bits2Linear(3MLIB)</code> , <code>mllib_SignalADPCM4Bits2Linear(3MLIB)</code> , <code>mllib_SignalADPCM5Bits2Linear(3MLIB)</code> , <code>mllib_SignalADPCMFree(3MLIB)</code> , <code>mllib_SignalADPCMInit(3MLIB)</code> , <code>mllib_SignalLinear2ADPCM2Bits(3MLIB)</code> , <code>mllib_SignalLinear2ADPCM3Bits(3MLIB)</code> , <code>mllib_SignalLinear2ADPCM4Bits(3MLIB)</code> , <code>mllib_SignalLinear2ADPCM5Bits(3MLIB)</code> , <code>attributes(5)</code>						

mlib_SignalADPCM4Bits2Linear(3MLIB)

NAME mlib_SignalADPCM4Bits2Linear – adaptive differential pulse code modulation (ADPCM)

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_SignalADPCM4Bits2Linear(mlib_s16 *pcm, const
    mlib_u8 *adpcm, void *state, mlib_s32 n);
```

DESCRIPTION The `mlib_SignalADPCM4Bits2Linear()` function performs adaptive differential pulse code modulation (ADPCM) in compliance with the ITU (former CCITT) G.721, G.723, and G.726 specifications. It converts data from G.721 or G.726 32kbps 4-bit ADPCM to 16-bit linear PCM format.

PARAMETERS The function takes the following arguments:

pcm Linear PCM sample array.

adpcm ADPCM code array.

state Internal structure of the codec.

n Number of samples in the source array.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mlib_SignalADPCM2Bits2Linear(3MLIB)`,
`mlib_SignalADPCM3Bits2Linear(3MLIB)`,
`mlib_SignalADPCM5Bits2Linear(3MLIB)`, `mlib_SignalADPCMFree(3MLIB)`,
`mlib_SignalADPCMInit(3MLIB)`, `mlib_SignalLinear2ADPCM2Bits(3MLIB)`,
`mlib_SignalLinear2ADPCM3Bits(3MLIB)`,
`mlib_SignalLinear2ADPCM4Bits(3MLIB)`,
`mlib_SignalLinear2ADPCM5Bits(3MLIB)`, `attributes(5)`

mllib_SignalADPCM5Bits2Linear(3MLIB)

NAME	mllib_SignalADPCM5Bits2Linear – adaptive differential pulse code modulation (ADPCM)						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalADPCM5Bits2Linear(mllib_s16 *<i>pcm</i>, const mllib_u8 *<i>adpcm</i>, void *<i>state</i>, mllib_s32 <i>n</i>);</pre>						
DESCRIPTION	The <code>mllib_SignalADPCM5Bits2Linear()</code> function performs adaptive differential pulse code modulation (ADPCM) in compliance with the ITU (former CCITT) G.721, G.723, and G.726 specifications. It converts data from G.723 or G.726 40kbps 5-bit ADPCM to 16-bit linear PCM format.						
PARAMETERS	The function takes the following arguments: <i>pcm</i> Linear PCM sample array. <i>adpcm</i> ADPCM code array. <i>state</i> Internal structure of the codec. <i>n</i> Number of samples in the source array.						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_SignalADPCM2Bits2Linear(3MLIB)</code> , <code>mllib_SignalADPCM3Bits2Linear(3MLIB)</code> , <code>mllib_SignalADPCM4Bits2Linear(3MLIB)</code> , <code>mllib_SignalADPCMFree(3MLIB)</code> , <code>mllib_SignalADPCMInit(3MLIB)</code> , <code>mllib_SignalLinear2ADPCM2Bits(3MLIB)</code> , <code>mllib_SignalLinear2ADPCM3Bits(3MLIB)</code> , <code>mllib_SignalLinear2ADPCM4Bits(3MLIB)</code> , <code>mllib_SignalLinear2ADPCM5Bits(3MLIB)</code> , <code>attributes(5)</code>						

NAME | mllib_SignalADPCMFree – adaptive differential pulse code modulation (ADPCM)

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_SignalADPCMFree(void *state);
```

DESCRIPTION | The mllib_SignalADPCMFree() function frees the internal structure for the codec for functions that perform adaptive differential pulse code modulation (ADPCM) in compliance with the ITU (former CCITT) G.721, G.723, and G.726 specifications.

PARAMETERS | The function takes the following arguments:
state Internal structure of the codec.

RETURN VALUES | None.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_SignalADPCM2Bits2Linear(3MLIB),
mllib_SignalADPCM3Bits2Linear(3MLIB),
mllib_SignalADPCM4Bits2Linear(3MLIB),
mllib_SignalADPCM5Bits2Linear(3MLIB), mllib_SignalADPCMInit(3MLIB),
mllib_SignalLinear2ADPCM2Bits(3MLIB),
mllib_SignalLinear2ADPCM3Bits(3MLIB),
mllib_SignalLinear2ADPCM4Bits(3MLIB),
mllib_SignalLinear2ADPCM5Bits(3MLIB), attributes(5)

mllib_SignalADPCMInit(3MLIB)

NAME	mllib_SignalADPCMInit – adaptive differential pulse code modulation (ADPCM)
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalADPCMInit(void **state);</pre>
DESCRIPTION	The mllib_SignalADPCMInit() function creates the internal structure for the codec for functions that perform adaptive differential pulse code modulation (ADPCM) in compliance with the ITU (former CCITT) G.721, G.723, and G.726 specifications.
PARAMETERS	The function takes the following arguments: <i>state</i> Internal structure of the codec.
RETURN VALUES	The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalADPCM2Bits2Linear(3MLIB),
mllib_SignalADPCM3Bits2Linear(3MLIB),
mllib_SignalADPCM4Bits2Linear(3MLIB),
mllib_SignalADPCM5Bits2Linear(3MLIB), mllib_SignalADPCMFree(3MLIB),
mllib_SignalLinear2ADPCM2Bits(3MLIB),
mllib_SignalLinear2ADPCM3Bits(3MLIB),
mllib_SignalLinear2ADPCM4Bits(3MLIB),
mllib_SignalLinear2ADPCM5Bits(3MLIB), attributes(5)

mlib_SignalALaw2Linear(3MLIB)

NAME mlib_SignalALaw2Linear – ITU G.711 m-law and A-law compression and decompression

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_SignalALaw2Linear(mlib_s16 *pcm, const mlib_u8
    *acode, mlib_s32 n);
```

DESCRIPTION The `mlib_SignalALaw2Linear()` function performs ITU G.711 m-law and A-law compression and decompression in compliance with the ITU (Former CCITT) G.711 specification.

PARAMETERS The function takes the following arguments:

pcm Linear PCM sample array.

acode A-law code array.

n Number of samples in the source array.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mlib_SignalALaw2uLaw(3MLIB)`, `mlib_SignalLinear2ALaw(3MLIB)`, `mlib_SignalLinear2uLaw(3MLIB)`, `mlib_SignaluLaw2ALaw(3MLIB)`, `mlib_SignaluLaw2Linear(3MLIB)`, `attributes(5)`

mllib_SignalALaw2uLaw(3MLIB)

NAME mllib_SignalALaw2uLaw – ITU G.711 m-law and A-law compression and decompression

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalALaw2uLaw(mllib_u8 *ucode, const mllib_u8
    *acode, mllib_s32 n);
```

DESCRIPTION The mllib_SignalALaw2uLaw() function performs ITU G.711 m-law and A-law compression and decompression in compliance with the ITU (Former CCITT) G.711 specification.

PARAMETERS The function takes the following arguments:

ucode m-law code array.

acode A-law code array.

n Number of samples in the source array.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalALaw2Linear(3MLIB), mllib_SignalLinear2ALaw(3MLIB), mllib_SignalLinear2uLaw(3MLIB), mllib_SignaluLaw2ALaw(3MLIB), mllib_SignaluLaw2Linear(3MLIB), attributes(5)

NAME	mllib_SignalAutoCorrel_S16, mllib_SignalAutoCorrel_S16S, mllib_SignalAutoCorrel_F32, mllib_SignalAutoCorrel_F32S – signal auto-correlation
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalAutoCorrel_S16(mllib_d64 *correl, const mllib_s16 *src, mllib_s32 disp, mllib_s32 n); mllib_status mllib_SignalAutoCorrel_S16S(mllib_d64 *correl, const mllib_s16 *src, mllib_s32 disp, mllib_s32 n); mllib_status mllib_SignalAutoCorrel_F32(mllib_d64 *correl, const mllib_f32 *src, mllib_s32 disp, mllib_s32 n); mllib_status mllib_SignalAutoCorrel_F32S(mllib_d64 *correl, const mllib_f32 *src, mllib_s32 disp, mllib_s32 n);</pre>
DESCRIPTION	<p>Each of these functions performs auto-correlation.</p> <p>For monaural signals, the following equation is used:</p> $\text{correl}[0] = \frac{1}{n-d} * \text{SUM}_{i=0}^{n-d-1} (\text{src}[i] * \text{src}[i+d])$ <p>For stereo signals, the following equation is used:</p> $\text{correl}[0] = \frac{1}{n-d} * \text{SUM}_{i=0}^{n-d-1} (\text{src}[2*i] * \text{src}[2*(i+d)])$ $\text{correl}[1] = \frac{1}{n-d} * \text{SUM}_{i=0}^{n-d-1} (\text{src}[2*i+1] * \text{src}[2*(i+d)+1])$ <p>where $d = \text{disp}$.</p>
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <p><i>correl</i> Pointer to the auto-correlation array. In the stereo version, <i>correl</i>[0] contains the auto-correlation of channel 0, and <i>correl</i>[1] contains the auto-correlation of channel 1.</p> <p><i>src</i> Source signal array.</p> <p><i>disp</i> Displacement. $0 \leq \text{disp} < n$.</p> <p><i>n</i> Number of samples in the source signal array.</p>
RETURN VALUES	Each of the functions returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .

`mlib_SignalAutoCorrel_S16(3MLIB)`

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mlib_SignalCrossCorrel_S16(3MLIB)`, `attributes(5)`

NAME mllib_SignalCepstral_F32 – perform cepstral analysis

SYNOPSIS `cc [flag...] file... -lmllib [library...]
#include <mllib.h>

mllib_status mllib_SignalCepstral_F32(mllib_f32 *cepst, const
mllib_f32 *signal, void *state);`

DESCRIPTION The mllib_SignalCepstral_F32() function performs cepstral analysis.

The basic operations to compute the cepstrum is shown below.

```

+-----+      +-----+      +-----+
| Fourier |      | log|*|      | Inverse |
----->|----->|----->|----->
x(n) | Transform | X(k) |      | X'(k) | Transform | c(n)
+-----+      +-----+      +-----+

```

where $x(n)$ is the input signal and $c(n)$ is its cepstrum. In mathematics, they are

$$X(k) = \sum_{n=0}^{N-1} x(n) * \exp(-j * \frac{2 * \text{PI} * k * n}{N})$$

$$X'(k) = \log |X(k)|$$

$$c(n) = \frac{1}{N} \sum_{k=0}^{N-1} X'(k) * \exp(j * \frac{2 * \text{PI} * k * n}{N})$$

Since $X'(k)$ is real and even (symmetric), i.e.

$$X'(k) = X'(N - k)$$

the $c(n)$ is real and the equation becomes Cosine transform.

$$c(n) = \frac{1}{N} \sum_{k=0}^{N-1} X'(k) * \cos(\frac{2 * \text{PI} * k * n}{N})$$

The cepstral coefficients in LPC is a special case of the above.

See *Digital Signal Processing* by Alan V. Oppenheim and Ronald W. Schafer, Prentice Hall, 1974.

See *Fundamentals of Speech Recognition* by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.

PARAMETERS The function takes the following arguments:

cepst The cepstral coefficients.

signal The input signal vector.

state Pointer to the internal state structure.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

mllib_SignalCepstral_F32(3MLIB)

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_SignalCepstralInit_F32(3MLIB)`,
`mllib_SignalCepstralFree_F32(3MLIB)`, `attributes(5)`

NAME mllib_SignalCepstralFree_S16, mllib_SignalCepstralFree_F32 – clean up for cepstral analysis

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_SignalCepstralFree_S16(void *state) ;
void mllib_SignalCepstralFree_F32(void *state) ;
```

DESCRIPTION Each of these functions frees the internal state structure for cepstral analysis. This function cleans up the internal state structure and releases all memory buffers.

PARAMETERS Each of the functions takes the following arguments:
state Pointer to the internal state structure.

RETURN VALUES None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalCepstral_S16(3MLIB), mllib_SignalCepstral_F32(3MLIB), mllib_SignalCepstral_S16_Adp(3MLIB), mllib_SignalCepstralInit_S16(3MLIB), mllib_SignalCepstralInit_F32(3MLIB), attributes(5)

mllib_SignalCepstralInit_S16(3MLIB)

NAME	mllib_SignalCepstralInit_S16, mllib_SignalCepstralInit_F32 – initialization for cepstral analysis						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalCepstralInit_S16(void *state, mllib_s32 order); mllib_status mllib_SignalCepstralInit_F32(void *state, mllib_s32 order);</pre>						
DESCRIPTION	<p>Each of these functions initializes the internal state structure for cepstral analysis.</p> <p>The init function performs internal state structure allocation and global initialization. Per function call initialization is done in each function, so the same internal state structure can be reused for multiple function calls.</p>						
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <p><i>order</i> The order of the input signal vector and the cepstral coefficients, where length = 2**order.</p> <p><i>state</i> Pointer to the internal state structure.</p>						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_SignalCepstral_S16(3MLIB), mllib_SignalCepstral_S16_Adp(3MLIB), mllib_SignalCepstral_F32(3MLIB), mllib_SignalCepstralFree_S16(3MLIB), mllib_SignalCepstralFree_F32(3MLIB), attributes(5)						

NAME mllib_SignalCepstral_S16 – perform cepstral analysis

SYNOPSIS `cc [flag...] file... -lmllib [library...]
#include <mllib.h>`

`mllib_status mllib_SignalCepstral_S16(mllib_s16 *cepst, mllib_s32
cscale, const mllib_s16 *signal, void *state);`

DESCRIPTION The `mllib_SignalCepstral_S16()` function performs cepstral analysis. The user supplied scaling factor will be used and the output will be saturated if necessary.

The basic operations to compute the cepstrum is shown below.

```

+-----+ +-----+ +-----+
| Fourier | |         | | Inverse |
----->|         |----->| log|*|  |----->| Fourier |----->
x(n) | Transform | X(k) | |         | X'(k) | Transform | c(n)
+-----+ +-----+ +-----+

```

where $x(n)$ is the input signal and $c(n)$ is its cepstrum. In mathematics, they are

$$X(k) = \sum_{n=0}^{N-1} x(n) * \exp(-j * \frac{2 * \pi * k * n}{N})$$

$$X'(k) = \log |X(k)|$$

$$c(n) = \frac{1}{N} \sum_{k=0}^{N-1} X'(k) * \exp(j * \frac{2 * \pi * k * n}{N})$$

Since $X'(k)$ is real and even (symmetric), i.e.

$$X'(k) = X'(N - k)$$

the $c(n)$ is real and the equation becomes Cosine transform.

$$c(n) = \frac{1}{N} \sum_{k=0}^{N-1} X'(k) * \cos(\frac{2 * \pi * k * n}{N})$$

The cepstral coefficients in LPC is a special case of the above.

See *Digital Signal Processing* by Alan V. Oppenheim and Ronald W. Schaffer, Prentice Hall, 1974.

See *Fundamentals of Speech Recognition* by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.

PARAMETERS The function takes the following arguments:

cepst The cepstral coefficients.

cscale The scaling factor of cepstral coefficients, where `actual_data = output_data * 2**(-scaling_factor)`.

signal The input signal vector, the signal samples are in Q15 format.

mllib_SignalCepstral_S16(3MLIB)

state Pointer to the internal state structure.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalCepstralInit_S16(3MLIB)`,
`mllib_SignalCepstral_S16_Adp(3MLIB)`,
`mllib_SignalCepstralFree_S16(3MLIB)`, `attributes(5)`

NAME | mllib_SignalCepstral_S16_AdP – perform cepstral analysis

SYNOPSIS | `cc [flag...] file... -lmllib [library...]`
`#include <mllib.h>`

`mllib_status mllib_SignalCepstral_S16_AdP(mllib_s16 *cepst, mllib_s32 *cscale, const mllib_s16 *signal, void *state);`

DESCRIPTION | The `mllib_SignalCepstral_S16_AdP()` function performs cepstral analysis. The scaling factor of the output data will be calculated based on the actual data.

The basic operations to compute the cepstrum is shown below.

```

+-----+      +-----+      +-----+
| Fourier |      | log|*|      | Inverse |
----->|----->|----->|----->|----->
x(n) | Transform | X(k) | | X'(k) | Transform | c(n)
+-----+      +-----+      +-----+

```

where $x(n)$ is the input signal and $c(n)$ is its cepstrum. In mathematics, they are

$$X(k) = \sum_{n=0}^{N-1} x(n) * \exp(-j * \frac{2 * \pi * k * n}{N})$$

$$X'(k) = \log |X(k)|$$

$$c(n) = \frac{1}{N} \sum_{k=0}^{N-1} X'(k) * \exp(j * \frac{2 * \pi * k * n}{N})$$

Since $X'(k)$ is real and even (symmetric), i.e.

$$X'(k) = X'(N - k)$$

the $c(n)$ is real and the equation becomes Cosine transform.

$$c(n) = \frac{1}{N} \sum_{k=0}^{N-1} X'(k) * \cos(\frac{2 * \pi * k * n}{N})$$

The cepstral coefficients in LPC is a special case of the above.

See *Digital Signal Processing* by Alan V. Oppenheim and Ronald W. Schaffer, Prentice Hall, 1974.

See *Fundamentals of Speech Recognition* by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.

PARAMETERS | The function takes the following arguments:

cepst | The cepstral coefficients.

cscale | The scaling factor of cepstral coefficients, where `actual_data = output_data * 2**(-scaling_factor)`.

signal | The input signal vector, the signal samples are in Q15 format.

`mllib_SignalCepstral_S16_Adp(3MLIB)`

state Pointer to the internal state structure.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalCepstralInit_S16(3MLIB)`, `mllib_SignalCepstral_S16(3MLIB)`, `mllib_SignalCepstralFree_S16(3MLIB)`, `attributes(5)`

mllib_SignalConvertShift_F32_U8(3MLIB)

NAME mllib_SignalConvertShift_F32_U8, mllib_SignalConvertShift_F32_S8, mllib_SignalConvertShift_F32_S16, mllib_SignalConvertShift_F32_S32, mllib_SignalConvertShift_F32S_U8S, mllib_SignalConvertShift_F32S_S8S, mllib_SignalConvertShift_F32S_S16S, mllib_SignalConvertShift_F32S_S32S – data type convert with shifting

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalConvertShift_F32_U8(mllib_f32 *dst, const
    mllib_u8 *src, mllib_s32 shift, mllib_s32 n);

mllib_status mllib_SignalConvertShift_F32_S8(mllib_f32 *dst, const
    mllib_s8 *src, mllib_s32 shift, mllib_s32 n);

mllib_status mllib_SignalConvertShift_F32_S16(mllib_f32 *dst, const
    mllib_s16 *src, mllib_s32 shift, mllib_s32 n);

mllib_status mllib_SignalConvertShift_F32_S32(mllib_f32 *dst, const
    mllib_s32 *src, mllib_s32 shift, mllib_s32 n);

mllib_status mllib_SignalConvertShift_F32S_U8S(mllib_f32 *dst, const
    mllib_u8 *src, mllib_s32 shift, mllib_s32 n);

mllib_status mllib_SignalConvertShift_F32S_S8S(mllib_f32 *dst, const
    mllib_s8 *src, mllib_s32 shift, mllib_s32 n);

mllib_status mllib_SignalConvertShift_F32S_S16S(mllib_f32 *dst,
    const mllib_s16 *src, mllib_s32 shift, mllib_s32 n);

mllib_status mllib_SignalConvertShift_F32S_S32S(mllib_f32 *dst,
    const mllib_s32 *src, mllib_s32 shift, mllib_s32 n);
```

DESCRIPTION Each of these functions performs data type convert with shifting.

The following equation is used:

$$dst[i] = src[i] * 2^{**shift}$$

See the following table for available variations of this group of data type convert functions.

Type [*]	F32	F32S
U8	Y	
S8	Y	
S16	Y	
S32	Y	
U8S		Y

mllib_SignalConvertShift_F32_U8(3MLIB)

Type [*]	F32	F32S
S8S		Y
S16S		Y
S32S		Y

[*] Each row represents a source data type. Each column represents a destination data type.

PARAMETERS Each of the functions takes the following arguments:

dst Destination signal array.
src Source signal array.
shift Left shifting factor.
n Number of samples in the source signal arrays.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalConvertShift_U8_S8_Sat(3MLIB)`, `attributes(5)`

mllib_SignalConvertShift_U8_S8_Sat(3MLIB)

NAME | mllib_SignalConvertShift_U8_S8_Sat, mllib_SignalConvertShift_U8_S16_Sat, mllib_SignalConvertShift_U8_S32_Sat, mllib_SignalConvertShift_U8_F32_Sat, mllib_SignalConvertShift_U8S_S8S_Sat, mllib_SignalConvertShift_U8S_S16S_Sat, mllib_SignalConvertShift_U8S_S32S_Sat, mllib_SignalConvertShift_U8S_F32S_Sat, mllib_SignalConvertShift_S8_U8_Sat, mllib_SignalConvertShift_S8_S16_Sat, mllib_SignalConvertShift_S8_S32_Sat, mllib_SignalConvertShift_S8_F32_Sat, mllib_SignalConvertShift_S8S_U8S_Sat, mllib_SignalConvertShift_S8S_S16S_Sat, mllib_SignalConvertShift_S8S_S32S_Sat, mllib_SignalConvertShift_S8S_F32S_Sat, mllib_SignalConvertShift_S16_U8_Sat, mllib_SignalConvertShift_S16_S8_Sat, mllib_SignalConvertShift_S16_S32_Sat, mllib_SignalConvertShift_S16_F32_Sat, mllib_SignalConvertShift_S16S_U8S_Sat, mllib_SignalConvertShift_S16S_S8S_Sat, mllib_SignalConvertShift_S16S_S32S_Sat, mllib_SignalConvertShift_S16S_F32S_Sat, mllib_SignalConvertShift_S32_U8_Sat, mllib_SignalConvertShift_S32_S8_Sat, mllib_SignalConvertShift_S32_S16_Sat, mllib_SignalConvertShift_S32_F32_Sat, mllib_SignalConvertShift_S32S_U8S_Sat, mllib_SignalConvertShift_S32S_S8S_Sat, mllib_SignalConvertShift_S32S_S16S_Sat, mllib_SignalConvertShift_S32S_F32S_Sat – data type convert with shifting

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalConvertShift_U8_S8_Sat(mllib_u8 *dst, const
    mllib_s8 *src, mllib_s32 shift, mllib_s32 n);

mllib_status mllib_SignalConvertShift_U8_S16_Sat(mllib_u8 *dst,
    const mllib_s16 *src, mllib_s32 shift, mllib_s32 n);

mllib_status mllib_SignalConvertShift_U8_S32_Sat(mllib_u8 *dst,
    const mllib_s32 *src, mllib_s32 shift, mllib_s32 n);

mllib_status mllib_SignalConvertShift_U8_F32_Sat(mllib_u8 *dst,
    const mllib_f32 *src, mllib_s32 shift, mllib_s32 n);

mllib_status mllib_SignalConvertShift_U8S_S8S_Sat(mllib_u8 *dst,
    const mllib_s8 *src, mllib_s32 shift, mllib_s32 n);

mllib_status mllib_SignalConvertShift_U8S_S16S_Sat(mllib_u8 *dst,
    const mllib_s16 *src, mllib_s32 shift, mllib_s32 n);

mllib_status mllib_SignalConvertShift_U8S_S32S_Sat(mllib_u8 *dst,
    const mllib_s32 *src, mllib_s32 shift, mllib_s32 n);

mllib_status mllib_SignalConvertShift_U8S_F32S_Sat(mllib_u8 *dst,
    const mllib_f32 *src, mllib_s32 shift, mllib_s32 n);

mllib_status mllib_SignalConvertShift_S8_U8_Sat(mllib_s8 *dst, const
    mllib_u8 *src, mllib_s32 shift, mllib_s32 n);

mllib_status mllib_SignalConvertShift_S8_S16_Sat(mllib_s8 *dst,
    const mllib_s16 *src, mllib_s32 shift, mllib_s32 n);

mllib_status mllib_SignalConvertShift_S8_S32_Sat(mllib_s8 *dst,
    const mllib_s32 *src, mllib_s32 shift, mllib_s32 n);
```

mllib_SignalConvertShift_U8_S8_Sat(3MLIB)

```
mllib_status mllib_SignalConvertShift_S8_F32_Sat(mllib_s8 *dst,
    const mllib_f32 *src, mllib_s32 shift, mllib_s32 n);
mllib_status mllib_SignalConvertShift_S8S_U8S_Sat(mllib_s8 *dst,
    const mllib_u8 *src, mllib_s32 shift, mllib_s32 n);
mllib_status mllib_SignalConvertShift_S8S_S16S_Sat(mllib_s8 *dst,
    const mllib_s16 *src, mllib_s32 shift, mllib_s32 n);
mllib_status mllib_SignalConvertShift_S8S_S32S_Sat(mllib_s8 *dst,
    const mllib_s32 *src, mllib_s32 shift, mllib_s32 n);
mllib_status mllib_SignalConvertShift_S8S_F32S_Sat(mllib_s8 *dst,
    const mllib_f32 *src, mllib_s32 shift, mllib_s32 n);
mllib_status mllib_SignalConvertShift_S16_U8_Sat(mllib_s16 *dst,
    const mllib_u8 *src, mllib_s32 shift, mllib_s32 n);
mllib_status mllib_SignalConvertShift_S16_S8_Sat(mllib_s16 *dst,
    const mllib_s8 *src, mllib_s32 shift, mllib_s32 n);
mllib_status mllib_SignalConvertShift_S16_S32_Sat(mllib_s16 *dst,
    const mllib_s32 *src, mllib_s32 shift, mllib_s32 n);
mllib_status mllib_SignalConvertShift_S16_F32_Sat(mllib_s16 *dst,
    const mllib_f32 *src, mllib_s32 shift, mllib_s32 n);
mllib_status mllib_SignalConvertShift_S16S_U8S_Sat(mllib_s16 *dst,
    const mllib_u8 *src, mllib_s32 shift, mllib_s32 n);
mllib_status mllib_SignalConvertShift_S16S_S8S_Sat(mllib_s16 *dst,
    const mllib_s8 *src, mllib_s32 shift, mllib_s32 n);
mllib_status mllib_SignalConvertShift_S16S_S32S_Sat(mllib_s16 *dst,
    const mllib_s32 *src, mllib_s32 shift, mllib_s32 n);
mllib_status mllib_SignalConvertShift_S16S_F32S_Sat(mllib_s16 *dst,
    const mllib_f32 *src, mllib_s32 shift, mllib_s32 n);
mllib_status mllib_SignalConvertShift_S32_U8_Sat(mllib_s32 *dst,
    const mllib_u8 *src, mllib_s32 shift, mllib_s32 n);
mllib_status mllib_SignalConvertShift_S32_S8_Sat(mllib_s32 *dst,
    const mllib_s8 *src, mllib_s32 shift, mllib_s32 n);
mllib_status mllib_SignalConvertShift_S32_S16_Sat(mllib_s32 *dst,
    const mllib_s16 *src, mllib_s32 shift, mllib_s32 n);
mllib_status mllib_SignalConvertShift_S32_F32_Sat(mllib_s32 *dst,
    const mllib_f32 *src, mllib_s32 shift, mllib_s32 n);
mllib_status mllib_SignalConvertShift_S32S_U8S_Sat(mllib_s32 *dst,
    const mllib_u8 *src, mllib_s32 shift, mllib_s32 n);
mllib_status mllib_SignalConvertShift_S32S_S8S_Sat(mllib_s32 *dst,
    const mllib_s8 *src, mllib_s32 shift, mllib_s32 n);
```

mllib_SignalConvertShift_U8_S8_Sat(3MLIB)

```
mllib_status mllib_SignalConvertShift_S32S_S16S_Sat(mllib_s32 *dst,
    const mllib_s16 *src, mllib_s32 shift, mllib_s32 n);

mllib_status mllib_SignalConvertShift_S32S_F32S_Sat(mllib_s32 *dst,
    const mllib_f32 *src, mllib_s32 shift, mllib_s32 n);
```

DESCRIPTION

Each of these functions performs data type convert with shifting.

The following equation is used:

$$dst[i] = saturate(src[i] * 2^{**shift})$$

See the following tables for available variations of this group of data type convert functions.

Type [*]	U8	S8	S16	S32
U8		Y	Y	Y
S8	Y		Y	Y
S16	Y	Y		Y
S32	Y	Y	Y	
F32	Y	Y	Y	Y

Type [*]	U8S	S8S	S16S	S32S
U8S		Y	Y	Y
S8S	Y		Y	Y
S16S	Y	Y		Y
S32S	Y	Y	Y	
F32S	Y	Y	Y	Y

[*] Each row represents a source data type. Each column represents a destination data type.

PARAMETERS

Each of the functions takes the following arguments:

- dst* Destination signal array.
- src* Source signal array.
- shift* Left shifting factor.
- n* Number of samples in the source signal arrays.

RETURN VALUES

Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

`mlib_SignalConvertShift_U8_S8_Sat(3MLIB)`

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mlib_SignalConvertShift_F32_U8(3MLIB)`, `attributes(5)`

NAME	mllib_SignalConv_S16_S16_Sat, mllib_SignalConv_S16S_S16S_Sat, mllib_SignalConv_F32_F32, mllib_SignalConv_F32S_F32S – signal convolution
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalConv_S16_S16_Sat(mllib_s16 *dst, const mllib_s16 *src1, const mllib_s16 *src2, mllib_s32 m, mllib_s32 n); mllib_status mllib_SignalConv_S16S_S16S_Sat(mllib_s16 *dst, const mllib_s16 *src1, const mllib_s16 *src2, mllib_s32 m, mllib_s32 n); mllib_status mllib_SignalConv_F32_F32(mllib_f32 *dst, const mllib_f32 *src1, const mllib_f32 *src2, mllib_s32 m, mllib_s32 n); mllib_status mllib_SignalConv_F32S_F32S(mllib_f32 *dst, const mllib_f32 *src1, const mllib_f32 *src2, mllib_s32 m, mllib_s32 n);</pre>
DESCRIPTION	<p>Each of these functions performs convolution.</p> <p>For monaural signals, the following equation is used:</p> $dst[i] = \sum_{j=0}^{m-1} (src1[j] * src2[i - j]) \quad \text{if } m \leq n$ $dst[i] = \sum_{j=0}^{n-1} (src2[j] * src1[i - j]) \quad \text{if } m > n$ <p>where $i = 0, 1, \dots, (m + n - 2)$.</p> <p>For stereo signals, the following equation is used:</p> $dst[2*i] = \sum_{j=0}^{m-1} (src1[2*j] * src2[2*(i - j)])$ $dst[2*i + 1] = \sum_{j=0}^{m-1} (src1[2*j + 1] * src2[2*(i - j) + 1])$ <p>if $m \leq n$, or</p> $dst[2*i] = \sum_{j=0}^{n-1} (src2[2*j] * src1[2*(i - j)])$ $dst[2*i + 1] = \sum_{j=0}^{n-1} (src2[2*j + 1] * src1[2*(i - j) + 1])$ <p>if $m > n$; where $i = 0, 1, \dots, (m + n - 2)$.</p>
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <p><i>dst</i> Destination signal array.</p> <p><i>src1</i> First source signal array.</p>

`mllib_SignalConv_S16_S16_Sat(3MLIB)`

src2 Second source signal array.
m Number of samples in the first source signal array.
n Number of samples in the second source signal arrays.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `attributes(5)`

NAME	mlib_SignalCrossCorrel_S16, mlib_SignalCrossCorrel_S16S, mlib_SignalCrossCorrel_F32, mlib_SignalCrossCorrel_F32S – signal cross correlation
SYNOPSIS	<pre>cc [flag...] file... -lmlib [library...] #include <mlib.h> mlib_status mlib_SignalCrossCorrel_S16(mlib_d64 *correl, const mlib_s16 *src1, const mlib_s16 *src2, mlib_s32 n); mlib_status mlib_SignalCrossCorrel_S16S(mlib_d64 *correl, const mlib_s16 *src1, const mlib_s16 *src2, mlib_s32 n); mlib_status mlib_SignalCrossCorrel_F32(mlib_d64 *correl, const mlib_f32 *src1, const mlib_f32 *src2, mlib_s32 n); mlib_status mlib_SignalCrossCorrel_F32S(mlib_d64 *correl, const mlib_f32 *src1, const mlib_f32 *src2, mlib_s32 n);</pre>
DESCRIPTION	<p>Each of these functions performs cross correlation.</p> <p>For monaural signals, the following equation is used:</p> $\text{correl}[0] = \frac{1}{n} \sum_{i=0}^{n-1} (\text{src1}[i] * \text{src2}[i])$ <p>For stereo signals, the following equation is used:</p> $\text{correl}[0] = \frac{1}{n} \sum_{i=0}^{n-1} (\text{src1}[2*i] * \text{src2}[2*i])$ $\text{correl}[1] = \frac{1}{n} \sum_{i=0}^{n-1} (\text{src1}[2*i + 1] * \text{src2}[2*i + 1])$
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <p><i>correl</i> Pointer to the cross correlation array. In the stereo version, <i>correl</i>[0] contains the cross correlation of channel 0, and <i>correl</i>[1] contains the cross correlation of channel 1.</p> <p><i>src1</i> First source signal array.</p> <p><i>src2</i> Second source signal array.</p> <p><i>n</i> Number of samples in the source signal arrays.</p>
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:

`mllib_SignalCrossCorrel_S16(3MLIB)`

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_SignalAutoCorrel_S16\(3MLIB\)](#), [attributes\(5\)](#)

mllib_SignalDownSample_S16_S16(3MLIB)

NAME	mllib_SignalDownSample_S16_S16, mllib_SignalDownSample_S16S_S16S, mllib_SignalDownSample_F32_F32, mllib_SignalDownSample_F32S_F32S – signal downsampling
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalDownSample_S16_S16(mllib_s16 *dst, const mllib_s16 *src, mllib_s32 factor, mllib_s32 phase, mllib_s32 n); mllib_status mllib_SignalDownSample_S16S_S16S(mllib_s16 *dst, const mllib_s16 *src, mllib_s32 factor, mllib_s32 phase, mllib_s32 n); mllib_status mllib_SignalDownSample_F32_F32(mllib_f32 *dst, const mllib_f32 *src, mllib_s32 factor, mllib_s32 phase, mllib_s32 n); mllib_status mllib_SignalDownSample_F32S_F32S(mllib_f32 *dst, const mllib_f32 *src, mllib_s32 factor, mllib_s32 phase, mllib_s32 n);</pre>
DESCRIPTION	<p>Each of these functions performs downsampling.</p> <p>For monaural signals, the following equation is used:</p> $dst[i] = src[i*factor + phase]$ <p>where $i = 0, 1, \dots, (n - 1 - phase)/factor$.</p> <p>For stereo signals, the following equation is used:</p> $dst[2*i] = src[2*(i*factor + phase)]$ $dst[2*i + 1] = src[2*(i*factor + phase) + 1]$ <p>where $i = 0, 1, \dots, (n - 1 - phase)/factor$.</p>
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <p><i>dst</i> Output signal array.</p> <p><i>src</i> Input signal array.</p> <p><i>factor</i> Factor by which to downsample. $factor \geq 1$.</p> <p><i>phase</i> Parameter that determines relative position of an output value, within the input signal. $0 \leq phase < factor$.</p> <p><i>n</i> Number of samples in the input signal array.</p>
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:

`mlib_SignalDownSample_S16_S16(3MLIB)`

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mlib_SignalUpSample_S16_S16\(3MLIB\)](#), [attributes\(5\)](#)

NAME	mllib_SignalDTWKScalar_F32 – perform dynamic time warping for K-best paths on scalar data
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalDTWKScalar_F32(mllib_d64 *dist, const mllib_f32 *dobs, mllib_s32 lobs, void *state);</pre>
DESCRIPTION	<p>The <code>mllib_SignalDTWKScalar_F32()</code> function performs dynamic time warping for K-best paths on scalar data.</p> <p>Assume the reference data are</p> $r(y), y=1,2,\dots,N$ <p>and the observed data are</p> $o(x), x=1,2,\dots,M$ <p>the dynamic time warping is to find a mapping function (a path)</p> $p(i) = \{px(i), py(i)\}, i=1,2,\dots,Q$ <p>with the minimum distance.</p> <p>In K-best paths case, K paths with the K minimum distances are searched.</p> <p>The distance of a path is defined as</p> $\text{dist} = \sum_{i=1}^Q d(r(py(i)), o(px(i))) * m(px(i), py(i))$ <p>where $d(r, o)$ is the dissimilarity between data point/vector r and data point/vector o; $m(x, y)$ is the path weighting coefficient associated with path point (x, y); N is the length of the reference data; M is the length of the observed data; Q is the length of the path.</p> <p>Using L1 norm (sum of absolute differences)</p> $d(r, o) = \sum_{i=0}^{L-1} r(i) - o(i) $ <p>Using L2 norm (Euclidean distance)</p> $d(r, o) = \text{SQRT} \left\{ \sum_{i=0}^{L-1} (r(i) - o(i))^2 \right\}$ <p>where L is the length of each data vector.</p> <p>To scalar data where $L=1$, the two norms are the same.</p> $d(r, o) = r - o = \text{SQRT} \left\{ (r - o)^2 \right\}$ <p>The constraints of dynamic time warping are:</p>

1. Endpoint constraints

$px(1) = 1$
 $1 \leq py(1) \leq 1 + \text{delta}$
 and

$px(Q) = M$
 $N - \text{delta} \leq py(Q) \leq N$

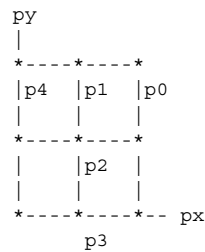
2. Monotonicity Conditions

$px(i) \leq px(i+1)$
 $py(i) \leq py(i+1)$

3. Local Continuity Constraints

See Table 4.5 on page 211 in Rabiner and Juang's book.

Itakura Type:



Allowable paths are

$p1 \rightarrow p0$ (1, 0)
 $p2 \rightarrow p0$ (1, 1)
 $p3 \rightarrow p0$ (1, 2)

Consecutive (1, 0) (1, 0) is disallowed. So path $p4 \rightarrow p1 \rightarrow p0$ is disallowed.

4. Global Path Constraints

Due to local continuity constraints, certain portions of the (px, py) plane are excluded from the region the optimal warping path can transverse. This forms global path constraints.

5. Slope Weighting

See Equation 4.150-3 on page 216 in Rabiner and Juang's book.

A path in (px, py) plane can be represented in chain code. The value of the chain code is defined as following.

```

=====
shift ( x , y ) | chain code
-----
( 1 , 0 ) | 0
( 0 , 1 ) | 1
( 1 , 1 ) | 2
( 2 , 1 ) | 3
( 1 , 2 ) | 4
( 3 , 1 ) | 5
( 3 , 2 ) | 6
( 1 , 3 ) | 7
    
```


`mllib_SignalDTWKScalar_F32(3MLIB)`

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalDTWKScalarInit_F32(3MLIB)`,
`mllib_SignalDTWKScalar_F32(3MLIB)`,
`mllib_SignalDTWKScalarPath_F32(3MLIB)`,
`mllib_SignalDTWKScalarFree_F32(3MLIB)`, `attributes(5)`

mllib_SignalDTWKScalarFree_S16(3MLIB)

NAME mllib_SignalDTWKScalarFree_S16, mllib_SignalDTWKScalarFree_F32 – clean up for K-best paths of scalar data

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_SignalDTWKScalarFree_S16(void *state) ;
void mllib_SignalDTWKScalarFree_F32(void *state) ;
```

DESCRIPTION Each of these functions frees the internal state structure for dynamic time warping (DTW) for K-best paths of scalar data.

This function cleans up the internal state structure and releases all memory buffers.

PARAMETERS Each of the functions takes the following arguments:

state Pointer to the internal state structure.

RETURN VALUES None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalDTWKScalarInit_S16(3MLIB),
mllib_SignalDTWKScalarInit_F32(3MLIB),
mllib_SignalDTWKScalar_S16(3MLIB), mllib_SignalDTWKScalar_F32(3MLIB),
mllib_SignalDTWKScalarPath_S16(3MLIB),
mllib_SignalDTWKScalarPath_F32(3MLIB), attributes(5)

mllib_SignalDTWKScalarInit_F32(3MLIB)

NAME	mllib_SignalDTWKScalarInit_F32 – initialization for K-best paths of scalar data														
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalDTWKScalarInit_F32(void *state, const mllib_f32 *dref, mllib_s32 lref, mllib_s32 kbest, mllib_s32 delta, mllib_s32 local, mllib_s32 slope);</pre>														
DESCRIPTION	<p>The <code>mllib_SignalDTWKScalarInit_F32()</code> function initializes the internal state structure for dynamic time warping (DTW) for K-best paths of scalar data.</p> <p>The <code>init</code> function performs internal state structure allocation and global initialization. Per DTW function call initialization is done in DTW function, so the same internal state structure can be reused for multiple DTW function calls.</p>														
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>dref</i></td><td>The reference data array.</td></tr><tr><td><i>lref</i></td><td>The length of the reference data array.</td></tr><tr><td><i>kbest</i></td><td>The number of the best paths evaluated.</td></tr><tr><td><i>delta</i></td><td>The delta in the endpoint constraints.</td></tr><tr><td><i>local</i></td><td>The type of the local continuity constraints. <code>MLIB_DTW_ITAKURA</code> for Itakura type constraints.</td></tr><tr><td><i>slope</i></td><td>The type of the slope weighting. <code>MLIB_DTW_NONE</code> for no slope weighting.</td></tr><tr><td><i>state</i></td><td>Pointer to the internal state structure.</td></tr></table>	<i>dref</i>	The reference data array.	<i>lref</i>	The length of the reference data array.	<i>kbest</i>	The number of the best paths evaluated.	<i>delta</i>	The delta in the endpoint constraints.	<i>local</i>	The type of the local continuity constraints. <code>MLIB_DTW_ITAKURA</code> for Itakura type constraints.	<i>slope</i>	The type of the slope weighting. <code>MLIB_DTW_NONE</code> for no slope weighting.	<i>state</i>	Pointer to the internal state structure.
<i>dref</i>	The reference data array.														
<i>lref</i>	The length of the reference data array.														
<i>kbest</i>	The number of the best paths evaluated.														
<i>delta</i>	The delta in the endpoint constraints.														
<i>local</i>	The type of the local continuity constraints. <code>MLIB_DTW_ITAKURA</code> for Itakura type constraints.														
<i>slope</i>	The type of the slope weighting. <code>MLIB_DTW_NONE</code> for no slope weighting.														
<i>state</i>	Pointer to the internal state structure.														
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .														
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:														
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe								
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														
MT-Level	MT-Safe														
SEE ALSO	<code>mllib_SignalDTWKScalarInit_F32(3MLIB)</code> , <code>mllib_SignalDTWKScalar_F32(3MLIB)</code> , <code>mllib_SignalDTWKScalarPath_F32(3MLIB)</code> , <code>mllib_SignalDTWKScalarFree_F32(3MLIB)</code> , <code>attributes(5)</code>														

mlib_SignalDTWKScalarInit_S16(3MLIB)

NAME mlib_SignalDTWKScalarInit_S16 – initialization for K-best paths of scalar data

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_SignalDTWKScalarInit_S16(void *state, const
    mlib_s16 *dref, mlib_s32 lref, mlib_s32 kbest, mlib_s32 sref,
    mlib_s32 delta, mlib_s32 local, mlib_s32 slope);
```

DESCRIPTION The mlib_SignalDTWKScalarInit_S16() function initializes the internal state structure for dynamic time warping (DTW) for K-best paths of scalar data.

The init function performs internal state structure allocation and global initialization. Per DTW function call initialization is done in DTW function, so the same internal state structure can be reused for multiple DTW function calls.

PARAMETERS The function takes the following arguments:

dref The reference data array.

lref The length of the reference data array.

kbest The number of the best paths evaluated.

sref The scaling factor of the reference data array, where `actual_data = input_data * 2**(-scaling_factor)`.

delta The delta in the endpoint constraints.

local The type of the local continuity constraints. `MLIB_DTW_ITAKURA` for Itakura type constraints.

slope The type of the slope weighting. `MLIB_DTW_NONE` for no slope weighting.

state Pointer to the internal state structure.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mlib_SignalDTWKScalarInit_S16(3MLIB)`,
`mlib_SignalDTWKScalar_S16(3MLIB)`,
`mlib_SignalDTWKScalarPath_S16(3MLIB)`,
`mlib_SignalDTWKScalarFree_S16(3MLIB)`, `attributes(5)`

mllib_SignalDTWKScalarPath_S16(3MLIB)

NAME	mllib_SignalDTWKScalarPath_S16, mllib_SignalDTWKScalarPath_F32 – return K-best path on scalar data
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalDTWKScalarPath_S16(mllib_s32 *path, mllib_s32 *lpath, mllib_s32 kpath, void *state); mllib_status mllib_SignalDTWKScalarPath_F32(mllib_s32 *path, mllib_s32 *lpath, mllib_s32 kpath, void *state);</pre>
DESCRIPTION	<p>Each of these functions returns K-best path on scalar data.</p> <p>Assume the reference data are</p> $r(y), y=1,2,\dots,N$ <p>and the observed data are</p> $o(x), x=1,2,\dots,M$ <p>the dynamic time warping is to find a mapping function (a path)</p> $p(i) = \{px(i), py(i)\}, i=1,2,\dots,Q$ <p>with the minimum distance.</p> <p>In K-best paths case, K paths with the K minimum distances are searched.</p> <p>The distance of a path is defined as</p> $\text{dist} = \sum_{i=1}^Q d(r(py(i)), o(px(i))) * m(px(i), py(i))$ <p>where $d(r, o)$ is the dissimilarity between data point/vector r and data point/vector o; $m(x, y)$ is the path weighting coefficient associated with path point (x, y); N is the length of the reference data; M is the length of the observed data; Q is the length of the path.</p> <p>Using L1 norm (sum of absolute differences)</p> $d(r, o) = \sum_{i=0}^{L-1} r(i) - o(i) $ <p>Using L2 norm (Euclidean distance)</p> $d(r, o) = \text{SQRT} \left\{ \sum_{i=0}^{L-1} (r(i) - o(i))^2 \right\}$ <p>where L is the length of each data vector.</p> <p>To scalar data where $L=1$, the two norms are the same.</p> $d(r, o) = r - o = \text{SQRT} \{ (r - o)^2 \}$

The constraints of dynamic time warping are:

1. Endpoint constraints

$$px(1) = 1$$

$$1 \leq py(1) \leq 1 + \delta$$

and

$$px(Q) = M$$

$$N - \delta \leq py(Q) \leq N$$

2. Monotonicity Conditions

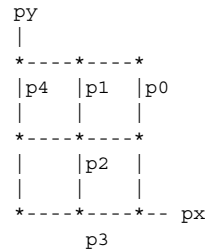
$$px(i) \leq px(i+1)$$

$$py(i) \leq py(i+1)$$

3. Local Continuity Constraints

See Table 4.5 on page 211 in Rabiner and Juang's book.

Itakura Type:



Allowable paths are

- p1->p0 (1, 0)
- p2->p0 (1, 1)
- p3->p0 (1, 2)

Consecutive (1, 0) (1, 0) is disallowed. So path p4->p1->p0 is disallowed.

4. Global Path Constraints

Due to local continuity constraints, certain portions of the (px, py) plane are excluded from the region the optimal warping path can transverse. This forms global path constraints.

5. Slope Weighting

See Equation 4.150-3 on page 216 in Rabiner and Juang's book.

A path in (px, py) plane can be represented in chain code. The value of the chain code is defined as following.

```

=====
shift ( x , y ) | chain code
-----
( 1 , 0 ) | 0
( 0 , 1 ) | 1
( 1 , 1 ) | 2
( 2 , 1 ) | 3
( 1 , 2 ) | 4
( 3 , 1 ) | 5
    
```

mllib_SignalDTWKScalarPath_S16(3MLIB)

```
( 3 , 2 ) | 6
( 1 , 3 ) | 7
( 2 , 3 ) | 8
=====
py
|
* 8 7 *
|
* 4 * 6
|
1 2 3 5
|
x--0--*--*-- px
```

where x marks the start point of a path segment, the numbers are the values of the chain code for the segment that ends at the point.

In following example, the observed data with 11 data points are mapped into the reference data with 9 data points

```
py
9 | * * * * * * * * * * * *
| * * * * * * * * * * * * /
| * * * * * * * * * * * * /
| * * * * * * * * * * * * /
| * * * * * * * * * * * * /
| * * * * * * * * * * * * /
| * * * * * * * * * * * * /
| * * * * * * * * * * * * /
| * * * * * * * * * * * * /
1 | * * * * * * * * * * * *
|
+----- px
1 11
```

The chain code that represents the path is

```
(2 2 2 1 2 0 2 2 0 2 0)
```

See *Fundamentals of Speech Recognition* by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.

- PARAMETERS**
- path* The optimal path.
 - lpath* The length of the optimal path.
 - kpath* The path index, $0 \leq kpath < kbest$.

`mllib_SignalDTWKScalarPath_S16(3MLIB)`

state Pointer to the internal state structure.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalDTWKScalarInit_S16(3MLIB)`,
`mllib_SignalDTWKScalarInit_F32(3MLIB)`,
`mllib_SignalDTWKScalar_S16(3MLIB)`, `mllib_SignalDTWKScalar_F32(3MLIB)`,
`mllib_SignalDTWKScalarFree_S16(3MLIB)`,
`mllib_SignalDTWKScalarFree_F32(3MLIB)`, `attributes(5)`

mllib_SignalDTWKScalar_S16(3MLIB)

NAME	mllib_SignalDTWKScalar_S16 – perform dynamic time warping for K-best paths on scalar data
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalDTWKScalar_S16(mllib_d64 *dist, const mllib_s16 *dobs, mllib_s32 lobs, mllib_s32 sob, void *state);</pre>
DESCRIPTION	<p>The mllib_SignalDTWKScalar_S16() function performs dynamic time warping for K-best paths on scalar data.</p> <p>Assume the reference data are</p> $r(y), y=1,2,\dots,N$ <p>and the observed data are</p> $o(x), x=1,2,\dots,M$ <p>the dynamic time warping is to find a mapping function (a path)</p> $p(i) = \{px(i), py(i)\}, i=1,2,\dots,Q$ <p>with the minimum distance.</p> <p>In K-best paths case, K paths with the K minimum distances are searched.</p> <p>The distance of a path is defined as</p> $\text{dist} = \sum_{i=1}^Q d(r(py(i)), o(px(i))) * m(px(i), py(i))$ <p>where $d(r, o)$ is the dissimilarity between data point/vector r and data point/vector o; $m(x, y)$ is the path weighting coefficient associated with path point (x, y); N is the length of the reference data; M is the length of the observed data; Q is the length of the path.</p> <p>Using L1 norm (sum of absolute differences)</p> $d(r, o) = \sum_{i=0}^{L-1} r(i) - o(i) $ <p>Using L2 norm (Euclidean distance)</p> $d(r, o) = \text{SQRT} \left\{ \sum_{i=0}^{L-1} (r(i) - o(i))^2 \right\}$ <p>where L is the length of each data vector.</p> <p>To scalar data where $L=1$, the two norms are the same.</p> $d(r, o) = r - o = \text{SQRT} \left\{ (r - o)^2 \right\}$ <p>The constraints of dynamic time warping are:</p>

1. Endpoint constraints

$$\begin{aligned} p_x(1) &= 1 \\ 1 \leq p_y(1) &\leq 1 + \text{delta} \end{aligned}$$

and

$$\begin{aligned} p_x(Q) &= M \\ N - \text{delta} \leq p_y(Q) &\leq N \end{aligned}$$

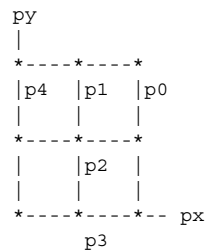
2. Monotonicity Conditions

$$\begin{aligned} p_x(i) &\leq p_x(i+1) \\ p_y(i) &\leq p_y(i+1) \end{aligned}$$

3. Local Continuity Constraints

See Table 4.5 on page 211 in Rabiner and Juang's book.

Itakura Type:



Allowable paths are

- p1->p0 (1, 0)
- p2->p0 (1, 1)
- p3->p0 (1, 2)

Consecutive (1, 0) (1, 0) is disallowed. So path p4->p1->p0 is disallowed.

4. Global Path Constraints

Due to local continuity constraints, certain portions of the (px, py) plane are excluded from the region the optimal warping path can transverse. This forms global path constraints.

5. Slope Weighting

See Equation 4.150-3 on page 216 in Rabiner and Juang's book.

A path in (px, py) plane can be represented in chain code. The value of the chain code is defined as following.

```

=====
shift ( x , y ) | chain code
-----
( 1 , 0 ) | 0
( 0 , 1 ) | 1
( 1 , 1 ) | 2
( 2 , 1 ) | 3
( 1 , 2 ) | 4
( 3 , 1 ) | 5
( 3 , 2 ) | 6
( 1 , 3 ) | 7
    
```

mllib_SignalDTWKScalar_S16(3MLIB)

```

      ( 2 , 3 ) |      8
      =====
      py
      |
      * 8 7 *
      |
      * 4 * 6
      |
      1 2 3 5
      |
      x--0--*--*-- px
  
```

where x marks the start point of a path segment, the numbers are the values of the chain code for the segment that ends at the point.

In following example, the observed data with 11 data points are mapped into the reference data with 9 data points

```

      py
      |
      9 * * * * * * * * * *
        | * * * * * * * * * *
        | * * * * * * * * * *
        | * * * * * * * * * *
        | * * * * * * * * * *
        | * * * * * * * * * *
        | * * * * * * * * * *
        | * * * * * * * * * *
        | * * * * * * * * * *
        | * * * * * * * * * *
        | * * * * * * * * * *
        | * * * * * * * * * *
        | * * * * * * * * * *
      1 * * * * * * * * * *
      |
      +----- px
      1                               11
  
```

The chain code that represents the path is

```
( 2 2 2 1 2 0 2 2 0 2 0 )
```

See *Fundamentals of Speech Recognition* by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.

PARAMETERS

The function takes the following arguments:

- dist* The distances of the K-best paths.
- dobs* The observed data array.
- lobs* The length of the observed data array.
- sobs* The scaling factor of the observed data array, where `actual_data = input_data * 2**(-scaling_factor)`.

`mllib_SignalDTWKScalar_S16(3MLIB)`

state Pointer to the internal state structure.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalDTWKScalarInit_S16(3MLIB)`,
`mllib_SignalDTWKScalar_S16(3MLIB)`,
`mllib_SignalDTWKScalarPath_S16(3MLIB)`,
`mllib_SignalDTWKScalarFree_S16(3MLIB)`, `attributes(5)`

mllib_SignalDTWKVector_F32(3MLIB)

NAME	mllib_SignalDTWKVector_F32 – perform dynamic time warping for K-best paths on vector data
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalDTWKVector_F32(mllib_d64 *dist, const mllib_f32 **dobs, mllib_s32 lobs, void *state);</pre>
DESCRIPTION	<p>The <code>mllib_SignalDTWKVector_F32()</code> function performs dynamic time warping for K-best paths on vector data.</p> <p>Assume the reference data are</p> $r(y), y=1,2,\dots,N$ <p>and the observed data are</p> $o(x), x=1,2,\dots,M$ <p>the dynamic time warping is to find a mapping function (a path)</p> $p(i) = \{px(i), py(i)\}, i=1,2,\dots,Q$ <p>with the minimum distance.</p> <p>In K-best paths case, K paths with the K minimum distances are searched.</p> <p>The distance of a path is defined as</p> $\text{dist} = \sum_{i=1}^Q d(r(py(i)), o(px(i))) * m(px(i), py(i))$ <p>where $d(r, o)$ is the dissimilarity between data point/vector r and data point/vector o; $m(x, y)$ is the path weighting coefficient associated with path point (x, y); N is the length of the reference data; M is the length of the observed data; Q is the length of the path.</p> <p>Using L1 norm (sum of absolute differences)</p> $d(r, o) = \sum_{i=0}^{L-1} r(i) - o(i) $ <p>Using L2 norm (Euclidean distance)</p> $d(r, o) = \text{SQRT} \left\{ \sum_{i=0}^{L-1} (r(i) - o(i))^2 \right\}$ <p>where L is the length of each data vector.</p> <p>To scalar data where $L=1$, the two norms are the same.</p> $d(r, o) = r - o = \text{SQRT} \left\{ (r - o)^2 \right\}$ <p>The constraints of dynamic time warping are:</p>

1. Endpoint constraints

$$px(1) = 1$$

$$1 \leq py(1) \leq 1 + \text{delta}$$

and

$$px(Q) = M$$

$$N - \text{delta} \leq py(Q) \leq N$$

2. Monotonicity Conditions

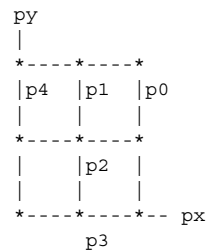
$$px(i) \leq px(i+1)$$

$$py(i) \leq py(i+1)$$

3. Local Continuity Constraints

See Table 4.5 on page 211 in Rabiner and Juang's book.

Itakura Type:



Allowable paths are

- p1->p0 (1, 0)
- p2->p0 (1, 1)
- p3->p0 (1, 2)

Consecutive (1, 0) (1, 0) is disallowed. So path p4->p1->p0 is disallowed.

4. Global Path Constraints

Due to local continuity constraints, certain portions of the (px, py) plane are excluded from the region the optimal warping path can transverse. This forms global path constraints.

5. Slope Weighting

See Equation 4.150-3 on page 216 in Rabiner and Juang's book.

A path in (px, py) plane can be represented in chain code. The value of the chain code is defined as following.

```

=====
shift ( x , y ) | chain code
-----
( 1 , 0 ) | 0
( 0 , 1 ) | 1
( 1 , 1 ) | 2
( 2 , 1 ) | 3
( 1 , 2 ) | 4
( 3 , 1 ) | 5
( 3 , 2 ) | 6
( 1 , 3 ) | 7
    
```

mllib_SignalDTWKVector_F32(3MLIB)

```

      ( 2 , 3 ) |      8
      =====
      py
      |
      * 8 7 *
      |
      * 4 * 6
      |
      1 2 3 5
      |
      x--0--*--*-- px
  
```

where x marks the start point of a path segment, the numbers are the values of the chain code for the segment that ends at the point.

In following example, the observed data with 11 data points are mapped into the reference data with 9 data points

```

      py
      |
      9 | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      1 | * * * * * * * * * *
      |
      +----- px
      1                               11
  
```

The chain code that represents the path is

```
( 2 2 2 1 2 0 2 2 0 2 0 )
```

See *Fundamentals of Speech Recognition* by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.

PARAMETERS

The function takes the following arguments:

- dist* The distances of the K-best paths.
- dobs* The observed data array.
- lobs* The length of the observed data array.
- state* Pointer to the internal state structure.

mllib_SignalDTWKVector_F32(3MLIB)

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalDTWKVectorInit_F32(3MLIB)`,
`mllib_SignalDTWKVector_F32(3MLIB)`,
`mllib_SignalDTWKVectorPath_F32(3MLIB)`,
`mllib_SignalDTWKVectorFree_F32(3MLIB)`, `attributes(5)`

mllib_SignalDTWKVectorFree_S16(3MLIB)

NAME mllib_SignalDTWKVectorFree_S16, mllib_SignalDTWKVectorFree_F32 – clean up for K-best paths of vector data

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_SignalDTWKVectorFree_S16(void *state);
void mllib_SignalDTWKVectorFree_F32(void *state);
```

DESCRIPTION Each of these functions frees the internal state structure for dynamic time warping (DTW) for K-best paths of vector data.

This function cleans up the internal state structure and releases all memory buffers.

PARAMETERS Each of the functions takes the following arguments:

state Pointer to the internal state structure.

RETURN VALUES None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalDTWKScalarInit_S16(3MLIB),
mllib_SignalDTWKVectorInit_F32(3MLIB),
mllib_SignalDTWKScalar_S16(3MLIB), mllib_SignalDTWKVector_F32(3MLIB),
mllib_SignalDTWKScalarPath_S16(3MLIB),
mllib_SignalDTWKVectorPath_F32(3MLIB), attributes(5)

mllib_SignalDTWKVectorInit_F32(3MLIB)

NAME | mllib_SignalDTWKVectorInit_F32 – initialization for K-best paths of vector data

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalDTWKVectorInit_F32(void *state, const
    mllib_f32 **dref, mllib_s32 lref, mllib_s32 ndata, mllib_s32 kbest,
    mllib_s32 dtype, mllib_s32 delta, mllib_s32 local, mllib_s32 slope);
```

DESCRIPTION | The `mllib_SignalDTWKVectorInit_F32()` function initializes the internal state structure for dynamic time warping (DTW) for K-best paths of vector data.

The `init` function performs internal state structure allocation and global initialization. Per DTW function call initialization is done in DTW function, so the same internal state structure can be reused for multiple DTW function calls.

PARAMETERS | The function takes the following arguments:

dref | The reference data array.

lref | The length of the reference data array.

ndata | The length of each data vector.

kbest | The number of the best paths evaluated.

dtype | The type of distance metric between data vectors. `MLIB_DTW_L1NORM` for L1 norm of difference (sum of absolute difference). `MLIB_DTW_L2NORM` for L2 norm of difference (Euclidean distance).

delta | The delta in the endpoint constraints.

local | The type of the local continuity constraints. `MLIB_DTW_ITAKURA` for Itakura type constraints.

slope | The type of the slope weighting. `MLIB_DTW_NONE` for no slope weighting.

state | Pointer to the internal state structure.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

`mllib_SignalDTWKVectorInit_F32(3MLIB)`

SEE ALSO | `mllib_SignalDTWKVectorInit_F32(3MLIB)`,
`mllib_SignalDTWKVector_F32(3MLIB)`,
`mllib_SignalDTWKVectorPath_F32(3MLIB)`,
`mllib_SignalDTWKVectorFree_F32(3MLIB)`, `attributes(5)`

mllib_SignalDTWKVectorInit_S16(3MLIB)

NAME mllib_SignalDTWKVectorInit_S16 – initialization for K-best paths of vector data

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalDTWKVectorInit_S16(void *state, const
    mllib_s16 **dref, mllib_s32 lref, mllib_s32 ndata, mllib_s32 kbest,
    mllib_s32 dtype, mllib_s32 sref, mllib_s32 delta, mllib_s32 local,
    mllib_s32 slope);
```

DESCRIPTION The mllib_SignalDTWKVectorInit_S16() function initializes the internal state structure for dynamic time warping (DTW) for K-best paths of vector data.

The init function performs internal state structure allocation and global initialization. Per DTW function call initialization is done in DTW function, so the same internal state structure can be reused for multiple DTW function calls.

PARAMETERS The function takes the following arguments:

dref The reference data array.

lref The length of the reference data array.

ndata The length of each data vector.

kbest The number of the best paths evaluated.

dtype The type of distance metric between data vectors. MLIB_DTW_L1NORM for L1 norm of difference (sum of absolute difference). MLIB_DTW_L2NORM for L2 norm of difference (Euclidean distance).

sref The scaling factor of the reference data array, where `actual_data = input_data * 2**(-scaling_factor)`.

delta The delta in the endpoint constraints.

local The type of the local continuity constraints. MLIB_DTW_ITAKURA for Itakura type constraints.

slope The type of the slope weighting. MLIB_DTW_NONE for no slope weighting.

state Pointer to the internal state structure.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

mllib_SignalDTWKVectorInit_S16(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	MT-Safe

SEE ALSO [mllib_SignalDTWKVectorInit_S16\(3MLIB\)](#),
[mllib_SignalDTWKVector_S16\(3MLIB\)](#),
[mllib_SignalDTWKVectorPath_S16\(3MLIB\)](#),
[mllib_SignalDTWKVectorFree_S16\(3MLIB\)](#), [attributes\(5\)](#)

NAME	mllib_SignalDTWKVectorPath_S16, mllib_SignalDTWKVectorPath_F32 – return K-best path on vector data
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalDTWKVectorPath_S16(mllib_s32 *path, mllib_s32 *lpath, mllib_s32 kpath, void *state); mllib_status mllib_SignalDTWKVectorPath_F32(mllib_s32 *path, mllib_s32 *lpath, mllib_s32 kpath, void *state);</pre>
DESCRIPTION	<p>Each of these functions returns K-best path on vector data.</p> <p>Assume the reference data are</p> $r(y), y=1,2,\dots,N$ <p>and the observed data are</p> $o(x), x=1,2,\dots,M$ <p>the dynamic time warping is to find a mapping function (a path)</p> $p(i) = \{px(i), py(i)\}, i=1,2,\dots,Q$ <p>with the minimum distance.</p> <p>In K-best paths case, K paths with the K minimum distances are searched.</p> <p>The distance of a path is defined as</p> $\text{dist} = \sum_{i=1}^Q d(r(py(i)), o(px(i))) * m(px(i), py(i))$ <p>where $d(r, o)$ is the dissimilarity between data point/vector r and data point/vector o; $m(x, y)$ is the path weighting coefficient associated with path point (x, y); N is the length of the reference data; M is the length of the observed data; Q is the length of the path.</p> <p>Using L1 norm (sum of absolute differences)</p> $d(r,o) = \sum_{i=0}^{L-1} r(i) - o(i) $ <p>Using L2 norm (Euclidean distance)</p> $d(r,o) = \text{SQRT} \{ \sum_{i=0}^{L-1} (r(i) - o(i))^{**2} \}$ <p>where L is the length of each data vector.</p> <p>To scalar data where $L=1$, the two norms are the same.</p> $d(r,o) = r - o = \text{SQRT} \{ (r - o)^{**2} \}$

The constraints of dynamic time warping are:

1. Endpoint constraints

$$\begin{aligned} p_x(1) &= 1 \\ 1 \leq p_y(1) &\leq 1 + \text{delta} \end{aligned}$$

and

$$\begin{aligned} p_x(Q) &= M \\ N - \text{delta} \leq p_y(Q) &\leq N \end{aligned}$$

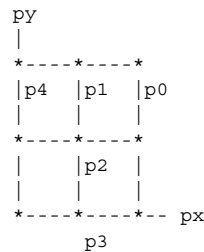
2. Monotonicity Conditions

$$\begin{aligned} p_x(i) &\leq p_x(i+1) \\ p_y(i) &\leq p_y(i+1) \end{aligned}$$

3. Local Continuity Constraints

See Table 4.5 on page 211 in Rabiner and Juang's book.

Itakura Type:



Allowable paths are

- p1->p0 (1, 0)
- p2->p0 (1, 1)
- p3->p0 (1, 2)

Consecutive (1, 0) (1, 0) is disallowed. So path p4->p1->p0 is disallowed.

4. Global Path Constraints

Due to local continuity constraints, certain portions of the (px, py) plane are excluded from the region the optimal warping path can transverse. This forms global path constraints.

5. Slope Weighting

See Equation 4.150-3 on page 216 in Rabiner and Juang's book.

A path in (px, py) plane can be represented in chain code. The value of the chain code is defined as following.

=====	
shift (x , y)	chain code

(1 , 0)	0
(0 , 1)	1
(1 , 1)	2
(2 , 1)	3
(1 , 2)	4
(3 , 1)	5

mllib_SignalDTWKVectorPath_S16(3MLIB)

state Pointer to the internal state structure.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalDTWKScalarInit_S16(3MLIB)`,
`mllib_SignalDTWKVectorInit_F32(3MLIB)`,
`mllib_SignalDTWKScalar_S16(3MLIB)`, `mllib_SignalDTWKVector_F32(3MLIB)`,
`mllib_SignalDTWKScalarFree_S16(3MLIB)`,
`mllib_SignalDTWKScalarFree_F32(3MLIB)`, `attributes(5)`

NAME	mllib_SignalDTWKVector_S16 – perform dynamic time warping for K-best paths on vector data
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalDTWKVector_S16(mllib_d64 *dist, const mllib_s16 **dobs, mllib_s32 lobs, mllib_s32 sob, void *state);</pre>
DESCRIPTION	<p>The <code>mllib_SignalDTWKVector_S16()</code> function performs dynamic time warping for K-best paths on vector data.</p> <p>Assume the reference data are</p> $r(y), y=1,2,\dots,N$ <p>and the observed data are</p> $o(x), x=1,2,\dots,M$ <p>the dynamic time warping is to find a mapping function (a path)</p> $p(i) = \{px(i), py(i)\}, i=1,2,\dots,Q$ <p>with the minimum distance.</p> <p>In K-best paths case, K paths with the K minimum distances are searched.</p> <p>The distance of a path is defined as</p> $\text{dist} = \sum_{i=1}^Q d(r(py(i)), o(px(i))) * m(px(i), py(i))$ <p>where $d(r, o)$ is the dissimilarity between data point/vector r and data point/vector o; $m(x, y)$ is the path weighting coefficient associated with path point (x, y); N is the length of the reference data; M is the length of the observed data; Q is the length of the path.</p> <p>Using L1 norm (sum of absolute differences)</p> $d(r, o) = \sum_{i=0}^{L-1} r(i) - o(i) $ <p>Using L2 norm (Euclidean distance)</p> $d(r, o) = \text{SQRT} \left\{ \sum_{i=0}^{L-1} (r(i) - o(i))^2 \right\}$ <p>where L is the length of each data vector.</p> <p>To scalar data where $L=1$, the two norms are the same.</p> $d(r, o) = r - o = \text{SQRT} \left\{ (r - o)^2 \right\}$ <p>The constraints of dynamic time warping are:</p>

1. Endpoint constraints

$$px(1) = 1$$

$$1 \leq py(1) \leq 1 + \text{delta}$$

and

$$px(Q) = M$$

$$N - \text{delta} \leq py(Q) \leq N$$

2. Monotonicity Conditions

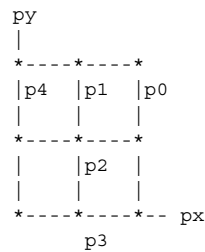
$$px(i) \leq px(i+1)$$

$$py(i) \leq py(i+1)$$

3. Local Continuity Constraints

See Table 4.5 on page 211 in Rabiner and Juang's book.

Itakura Type:



Allowable paths are

- p1->p0 (1,0)
- p2->p0 (1,1)
- p3->p0 (1,2)

Consecutive (1,0) (1,0) is disallowed. So path p4->p1->p0 is disallowed.

4. Global Path Constraints

Due to local continuity constraints, certain portions of the (px, py) plane are excluded from the region the optimal warping path can transverse. This forms global path constraints.

5. Slope Weighting

See Equation 4.150-3 on page 216 in Rabiner and Juang's book.

A path in (px, py) plane can be represented in chain code. The value of the chain code is defined as following.

=====	
shift (x , y)	chain code

(1 , 0)	0
(0 , 1)	1
(1 , 1)	2
(2 , 1)	3
(1 , 2)	4
(3 , 1)	5
(3 , 2)	6
(1 , 3)	7

`mllib_SignalDTWKVector_S16(3MLIB)`

state Pointer to the internal state structure.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalDTWKVectorInit_S16(3MLIB)`,
`mllib_SignalDTWKVector_S16(3MLIB)`,
`mllib_SignalDTWKVectorPath_S16(3MLIB)`,
`mllib_SignalDTWKVectorFree_S16(3MLIB)`, `attributes(5)`

NAME	mllib_SignalDTWScalar_F32 – perform dynamic time warping on scalar data
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalDTWScalar_F32(mllib_d64 *dist, const mllib_f32 *dobs, mllib_s32 lobs, void *state);</pre>
DESCRIPTION	<p>The mllib_SignalDTWScalar_F32() function performs dynamic time warping on scalar data.</p> <p>Assume the reference data are</p> $r(y), y=1,2,\dots,N$ <p>and the observed data are</p> $o(x), x=1,2,\dots,M$ <p>the dynamic time warping is to find a mapping function (a path)</p> $p(i) = \{px(i),py(i)\}, i=1,2,\dots,Q$ <p>with the minimum distance.</p> <p>In K-best paths case, K paths with the K minimum distances are searched.</p> <p>The distance of a path is defined as</p> $\text{dist} = \sum_{i=1}^Q d(r(py(i)), o(px(i))) * m(px(i), py(i))$ <p>where $d(r, o)$ is the dissimilarity between data point/vector r and data point/vector o; $m(x, y)$ is the path weighting coefficient associated with path point (x, y); N is the length of the reference data; M is the length of the observed data; Q is the length of the path.</p> <p>Using L1 norm (sum of absolute differences)</p> $d(r, o) = \sum_{i=0}^{L-1} r(i) - o(i) $ <p>Using L2 norm (Euclidean distance)</p> $d(r, o) = \text{SQRT} \{ \sum_{i=0}^{L-1} (r(i) - o(i))^{**2} \}$ <p>where L is the length of each data vector.</p> <p>To scalar data where $L=1$, the two norms are the same.</p> $d(r, o) = r - o = \text{SQRT} \{ (r - o)^{**2} \}$ <p>The constraints of dynamic time warping are:</p>

1. Endpoint constraints

$$px(1) = 1$$

$$1 \leq py(1) \leq 1 + \text{delta}$$

and

$$px(Q) = M$$

$$N - \text{delta} \leq py(Q) \leq N$$

2. Monotonicity Conditions

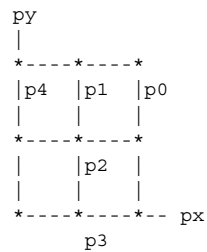
$$px(i) \leq px(i+1)$$

$$py(i) \leq py(i+1)$$

3. Local Continuity Constraints

See Table 4.5 on page 211 in Rabiner and Juang's book.

Itakura Type:



Allowable paths are

$$p1 \rightarrow p0 \quad (1, 0)$$

$$p2 \rightarrow p0 \quad (1, 1)$$

$$p3 \rightarrow p0 \quad (1, 2)$$

Consecutive (1, 0) (1, 0) is disallowed. So path p4 -> p1 -> p0 is disallowed.

4. Global Path Constraints

Due to local continuity constraints, certain portions of the (px, py) plane are excluded from the region the optimal warping path can transverse. This forms global path constraints.

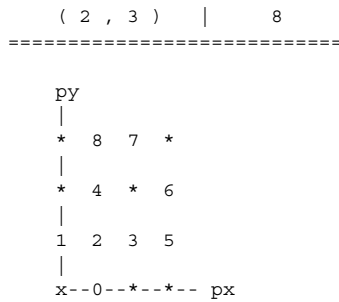
5. Slope Weighting

See Equation 4.150-3 on page 216 in Rabiner and Juang's book.

A path in (px, py) plane can be represented in chain code. The value of the chain code is defined as following.

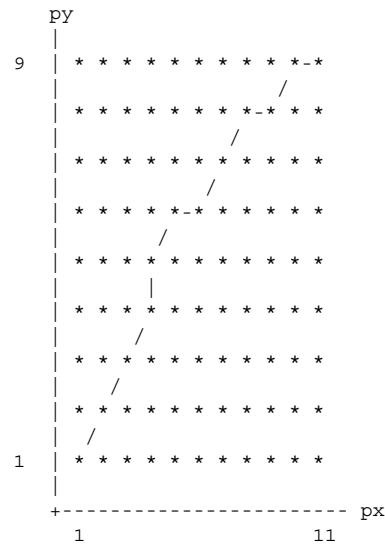
=====	
shift (x , y)	chain code

(1 , 0)	0
(0 , 1)	1
(1 , 1)	2
(2 , 1)	3
(1 , 2)	4
(3 , 1)	5
(3 , 2)	6
(1 , 3)	7



where x marks the start point of a path segment, the numbers are the values of the chain code for the segment that ends at the point.

In following example, the observed data with 11 data points are mapped into the reference data with 9 data points



The chain code that represents the path is

(2 2 2 1 2 0 2 2 0 2 0)

See *Fundamentals of Speech Recognition* by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.

PARAMETERS

The function takes the following arguments:

- dist* The distance of the optimal path.
- dobs* The observed data array.
- lobs* The length of the observed data array.
- state* Pointer to the internal state structure.

mllib_SignalDTWScalar_F32(3MLIB)

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalDTWScalarInit_F32(3MLIB)`,
`mllib_SignalDTWScalar_F32(3MLIB)`,
`mllib_SignalDTWScalarPath_F32(3MLIB)`,
`mllib_SignalDTWScalarFree_F32(3MLIB)`, `attributes(5)`

NAME mllib_SignalDTWScalarFree_S16, mllib_SignalDTWScalarFree_F32 – clean up for scalar data

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_SignalDTWScalarFree_S16(void *state);
void mllib_SignalDTWScalarFree_F32(void *state);
```

DESCRIPTION Each of these functions frees the internal state structure for dynamic time warping (DTW) of scalar data.

This function cleans up the internal state structure and releases all memory buffers.

PARAMETERS Each of the functions takes the following arguments:

state Pointer to the internal state structure.

RETURN VALUES None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalDTWScalarInit_S16(3MLIB),
mllib_SignalDTWScalarInit_F32(3MLIB),
mllib_SignalDTWScalar_S16(3MLIB), mllib_SignalDTWScalar_F32(3MLIB),
mllib_SignalDTWScalarPath_S16(3MLIB),
mllib_SignalDTWScalarPath_F32(3MLIB), attributes(5)

mllib_SignalDTWScalarInit_F32(3MLIB)

NAME	mllib_SignalDTWScalarInit_F32 – initialization for scalar data												
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalDTWScalarInit_F32(void *state, const mllib_f32 *dref, mllib_s32 lref, mllib_s32 delta, mllib_s32 local, mllib_s32 slope);</pre>												
DESCRIPTION	<p>The <code>mllib_SignalDTWScalarInit_F32()</code> function initializes the internal state structure for dynamic time warping (DTW) of scalar data.</p> <p>The <code>init</code> function performs internal state structure allocation and global initialization. Per DTW function call initialization is done in DTW function, so the same internal state structure can be reused for multiple DTW function calls.</p>												
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>dref</i></td><td>The reference data array.</td></tr><tr><td><i>lref</i></td><td>The length of the reference data array.</td></tr><tr><td><i>delta</i></td><td>The delta in the endpoint constraints.</td></tr><tr><td><i>local</i></td><td>The type of the local continuity constraints. <code>MLIB_DTW_ITAKURA</code> for Itakura type constraints.</td></tr><tr><td><i>slope</i></td><td>The type of the slope weighting. <code>MLIB_DTW_NONE</code> for no slope weighting.</td></tr><tr><td><i>state</i></td><td>Pointer to the internal state structure.</td></tr></table>	<i>dref</i>	The reference data array.	<i>lref</i>	The length of the reference data array.	<i>delta</i>	The delta in the endpoint constraints.	<i>local</i>	The type of the local continuity constraints. <code>MLIB_DTW_ITAKURA</code> for Itakura type constraints.	<i>slope</i>	The type of the slope weighting. <code>MLIB_DTW_NONE</code> for no slope weighting.	<i>state</i>	Pointer to the internal state structure.
<i>dref</i>	The reference data array.												
<i>lref</i>	The length of the reference data array.												
<i>delta</i>	The delta in the endpoint constraints.												
<i>local</i>	The type of the local continuity constraints. <code>MLIB_DTW_ITAKURA</code> for Itakura type constraints.												
<i>slope</i>	The type of the slope weighting. <code>MLIB_DTW_NONE</code> for no slope weighting.												
<i>state</i>	Pointer to the internal state structure.												
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .												
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:												
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe						
ATTRIBUTE TYPE	ATTRIBUTE VALUE												
Interface Stability	Evolving												
MT-Level	MT-Safe												
SEE ALSO	<code>mllib_SignalDTWScalarInit_F32(3MLIB)</code> , <code>mllib_SignalDTWScalar_F32(3MLIB)</code> , <code>mllib_SignalDTWScalarPath_F32(3MLIB)</code> , <code>mllib_SignalDTWScalarFree_F32(3MLIB)</code> , <code>attributes(5)</code>												

mllib_SignalDTWScalarInit_S16(3MLIB)

NAME | mllib_SignalDTWScalarInit_S16 – initialization for scalar data

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalDTWScalarInit_S16(void *state, const
      mllib_s16 *dref, mllib_s32 lref, mllib_s32 sref, mllib_s32 delta,
      mllib_s32 local, mllib_s32 slope);
```

DESCRIPTION | The mllib_SignalDTWScalarInit_S16() function initializes the internal state structure for dynamic time warping (DTW) of scalar data.

The init function performs internal state structure allocation and global initialization. Per DTW function call initialization is done in DTW function, so the same internal state structure can be reused for multiple DTW function calls.

PARAMETERS | The function takes the following arguments:

dref | The reference data array.

lref | The length of the reference data array.

sref | The scaling factor of the reference data array, where `actual_data = input_data * 2**(-scaling_factor)`.

delta | The delta in the endpoint constraints.

local | The type of the local continuity constraints. `MLIB_DTW_ITAKURA` for Itakura type constraints.

slope | The type of the slope weighting. `MLIB_DTW_NONE` for no slope weighting.

state | Pointer to the internal state structure.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_SignalDTWScalarInit_S16\(3MLIB\)](#),
[mllib_SignalDTWScalar_S16\(3MLIB\)](#),
[mllib_SignalDTWScalarPath_S16\(3MLIB\)](#),
[mllib_SignalDTWScalarFree_S16\(3MLIB\)](#), [attributes\(5\)](#)

mllib_SignalDTWScalarPath_F32(3MLIB)

NAME	mllib_SignalDTWScalarPath_F32 – perform dynamic time warping on scalar data
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalDTWScalarPath_F32(mllib_d64 *dist, mllib_s32 *path, mllib_s32 *lpath, const mllib_f32 *dobs, mllib_s32 lobs, void *state);</pre>
DESCRIPTION	<p>The mllib_SignalDTWScalarPath_F32() function performs dynamic time warping on scalar data.</p> <p>Assume the reference data are</p> $r(y), y=1,2,\dots,N$ <p>and the observed data are</p> $o(x), x=1,2,\dots,M$ <p>the dynamic time warping is to find a mapping function (a path)</p> $p(i) = \{px(i), py(i)\}, i=1,2,\dots,Q$ <p>with the minimum distance.</p> <p>In K-best paths case, K paths with the K minimum distances are searched.</p> <p>The distance of a path is defined as</p> $\text{dist} = \sum_{i=1}^Q d(r(py(i)), o(px(i))) * m(px(i), py(i))$ <p>where $d(r, o)$ is the dissimilarity between data point/vector r and data point/vector o; $m(x, y)$ is the path weighting coefficient associated with path point (x, y); N is the length of the reference data; M is the length of the observed data; Q is the length of the path.</p> <p>Using L1 norm (sum of absolute differences)</p> $d(r, o) = \sum_{i=0}^{L-1} r(i) - o(i) $ <p>Using L2 norm (Euclidean distance)</p> $d(r, o) = \text{SQRT} \left\{ \sum_{i=0}^{L-1} (r(i) - o(i))^2 \right\}$ <p>where L is the length of each data vector.</p> <p>To scalar data where $L=1$, the two norms are the same.</p> $d(r, o) = r - o = \text{SQRT} \left\{ (r - o)^2 \right\}$ <p>The constraints of dynamic time warping are:</p>

1. Endpoint constraints

$px(1) = 1$
 $1 \leq py(1) \leq 1 + \text{delta}$
 and

$px(Q) = M$
 $N - \text{delta} \leq py(Q) \leq N$

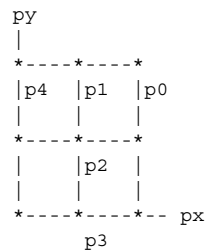
2. Monotonicity Conditions

$px(i) \leq px(i+1)$
 $py(i) \leq py(i+1)$

3. Local Continuity Constraints

See Table 4.5 on page 211 in Rabiner and Juang's book.

Itakura Type:



Allowable paths are

$p1 \rightarrow p0$ (1, 0)
 $p2 \rightarrow p0$ (1, 1)
 $p3 \rightarrow p0$ (1, 2)

Consecutive (1, 0) (1, 0) is disallowed. So path $p4 \rightarrow p1 \rightarrow p0$ is disallowed.

4. Global Path Constraints

Due to local continuity constraints, certain portions of the (px, py) plane are excluded from the region the optimal warping path can transverse. This forms global path constraints.

5. Slope Weighting

See Equation 4.150-3 on page 216 in Rabiner and Juang's book.

A path in (px, py) plane can be represented in chain code. The value of the chain code is defined as following.

```

=====
shift ( x , y ) | chain code
-----
( 1 , 0 ) | 0
( 0 , 1 ) | 1
( 1 , 1 ) | 2
( 2 , 1 ) | 3
( 1 , 2 ) | 4
( 3 , 1 ) | 5
( 3 , 2 ) | 6
( 1 , 3 ) | 7
    
```

mllib_SignalDTWScalarPath_F32(3MLIB)

```

      ( 2 , 3 ) |      8
      =====
      py
      |
      * 8 7 *
      |
      * 4 * 6
      |
      1 2 3 5
      |
      x--0--*--*-- px
  
```

where x marks the start point of a path segment, the numbers are the values of the chain code for the segment that ends at the point.

In following example, the observed data with 11 data points are mapped into the reference data with 9 data points

```

      py
      |
      9 | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      1 | * * * * * * * * * *
      |
      +----- px
      1                               11
  
```

The chain code that represents the path is

```
( 2 2 2 1 2 0 2 2 0 2 0 )
```

See *Fundamentals of Speech Recognition* by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.

PARAMETERS

The function takes the following arguments:

- dist* The distance of the optimal path.
- path* The optimal path.
- lpath* The length of the optimal path.
- dobs* The observed data array.

mllib_SignalDTWScalarPath_F32(3MLIB)

lobs The length of the observed data array.

state Pointer to the internal state structure.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalDTWScalarInit_F32(3MLIB)`,
`mllib_SignalDTWScalar_F32(3MLIB)`,
`mllib_SignalDTWScalarPath_F32(3MLIB)`,
`mllib_SignalDTWScalarFree_F32(3MLIB)`, `attributes(5)`

mllib_SignalDTWScalarPath_S16(3MLIB)

NAME	mllib_SignalDTWScalarPath_S16 – perform dynamic time warping on scalar data
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalDTWScalarPath_S16(mllib_d64 *dist, mllib_s32 *path, mllib_s32 *lpath, const mllib_s16 *dobs, mllib_s32 lobs, mllib_s32 sob, void *state);</pre>
DESCRIPTION	<p>The mllib_SignalDTWScalarPath_S16() function performs dynamic time warping on scalar data.</p> <p>Assume the reference data are</p> $r(y), y=1,2,\dots,N$ <p>and the observed data are</p> $o(x), x=1,2,\dots,M$ <p>the dynamic time warping is to find a mapping function (a path)</p> $p(i) = \{px(i), py(i)\}, i=1,2,\dots,Q$ <p>with the minimum distance.</p> <p>In K-best paths case, K paths with the K minimum distances are searched.</p> <p>The distance of a path is defined as</p> $\text{dist} = \sum_{i=1}^Q d(r(py(i)), o(px(i))) * m(px(i), py(i))$ <p>where $d(r, o)$ is the dissimilarity between data point/vector r and data point/vector o; $m(x, y)$ is the path weighting coefficient associated with path point (x, y); N is the length of the reference data; M is the length of the observed data; Q is the length of the path.</p> <p>Using L1 norm (sum of absolute differences)</p> $d(r, o) = \sum_{i=0}^{L-1} r(i) - o(i) $ <p>Using L2 norm (Euclidean distance)</p> $d(r, o) = \text{SQRT} \left\{ \sum_{i=0}^{L-1} (r(i) - o(i))^2 \right\}$ <p>where L is the length of each data vector.</p> <p>To scalar data where $L=1$, the two norms are the same.</p> $d(r, o) = r - o = \text{SQRT} \left\{ (r - o)^2 \right\}$ <p>The constraints of dynamic time warping are:</p>

1. Endpoint constraints

$px(1) = 1$
 $1 \leq py(1) \leq 1 + \text{delta}$
 and

$px(Q) = M$
 $N - \text{delta} \leq py(Q) \leq N$

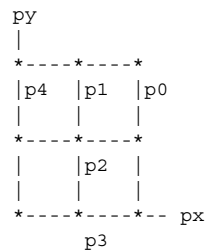
2. Monotonicity Conditions

$px(i) \leq px(i+1)$
 $py(i) \leq py(i+1)$

3. Local Continuity Constraints

See Table 4.5 on page 211 in Rabiner and Juang's book.

Itakura Type:



Allowable paths are

$p1 \rightarrow p0$ (1, 0)
 $p2 \rightarrow p0$ (1, 1)
 $p3 \rightarrow p0$ (1, 2)

Consecutive (1, 0) (1, 0) is disallowed. So path $p4 \rightarrow p1 \rightarrow p0$ is disallowed.

4. Global Path Constraints

Due to local continuity constraints, certain portions of the (px, py) plane are excluded from the region the optimal warping path can transverse. This forms global path constraints.

5. Slope Weighting

See Equation 4.150-3 on page 216 in Rabiner and Juang's book.

A path in (px, py) plane can be represented in chain code. The value of the chain code is defined as following.

```

=====
shift ( x , y ) | chain code
-----
( 1 , 0 ) | 0
( 0 , 1 ) | 1
( 1 , 1 ) | 2
( 2 , 1 ) | 3
( 1 , 2 ) | 4
( 3 , 1 ) | 5
( 3 , 2 ) | 6
( 1 , 3 ) | 7
    
```

mllib_SignalDTWScalarPath_S16(3MLIB)

```

      ( 2 , 3 ) |      8
      =====
      py
      |
      * 8 7 *
      |
      * 4 * 6
      |
      1 2 3 5
      |
      x--0--*--*-- px
  
```

where x marks the start point of a path segment, the numbers are the values of the chain code for the segment that ends at the point.

In following example, the observed data with 11 data points are mapped into the reference data with 9 data points

```

      py
      |
      9 | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      1 | * * * * * * * * * *
      |
      +-----+----- px
          1                11
  
```

The chain code that represents the path is

```
(2 2 2 1 2 0 2 2 0 2 0)
```

See *Fundamentals of Speech Recognition* by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.

PARAMETERS

The function takes the following arguments:

- dist* The distance of the optimal path.
- path* The optimal path.
- lpath* The length of the optimal path.
- dobs* The observed data array.

`mllib_SignalDTWScalarPath_S16(3MLIB)`

lobs The length of the observed data array.
sobs The scaling factor of the observed data array, where `actual_data = input_data * 2**(-scaling_factor)`.
state Pointer to the internal state structure.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalDTWScalarInit_S16(3MLIB)`,
`mllib_SignalDTWScalar_S16(3MLIB)`,
`mllib_SignalDTWScalarPath_S16(3MLIB)`,
`mllib_SignalDTWScalarFree_S16(3MLIB)`, `attributes(5)`

mllib_SignalDTWScalar_S16(3MLIB)

NAME	mllib_SignalDTWScalar_S16 – perform dynamic time warping on scalar data
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalDTWScalar_S16(mllib_d64 *dist, const mllib_s16 *dobs, mllib_s32 lobs, mllib_s32 sob, void *state);</pre>
DESCRIPTION	<p>The <code>mllib_SignalDTWScalar_S16()</code> function performs dynamic time warping on scalar data.</p> <p>Assume the reference data are</p> $r(y), y=1,2,\dots,N$ <p>and the observed data are</p> $o(x), x=1,2,\dots,M$ <p>the dynamic time warping is to find a mapping function (a path)</p> $p(i) = \{px(i),py(i)\}, i=1,2,\dots,Q$ <p>with the minimum distance.</p> <p>In K-best paths case, K paths with the K minimum distances are searched.</p> <p>The distance of a path is defined as</p> $\text{dist} = \sum_{i=1}^Q d(r(py(i)),o(px(i))) * m(px(i),py(i))$ <p>where $d(r, o)$ is the dissimilarity between data point/vector r and data point/vector o; $m(x, y)$ is the path weighting coefficient associated with path point (x, y); N is the length of the reference data; M is the length of the observed data; Q is the length of the path.</p> <p>Using L1 norm (sum of absolute differences)</p> $d(r,o) = \sum_{i=0}^{L-1} r(i) - o(i) $ <p>Using L2 norm (Euclidean distance)</p> $d(r,o) = \text{SQRT} \{ \sum_{i=0}^{L-1} (r(i) - o(i))^{**2} \}$ <p>where L is the length of each data vector.</p> <p>To scalar data where $L=1$, the two norms are the same.</p> $d(r,o) = r - o = \text{SQRT} \{ (r - o)^{**2} \}$ <p>The constraints of dynamic time warping are:</p>

1. Endpoint constraints

$px(1) = 1$
 $1 \leq py(1) \leq 1 + \text{delta}$
 and

$px(Q) = M$
 $N - \text{delta} \leq py(Q) \leq N$

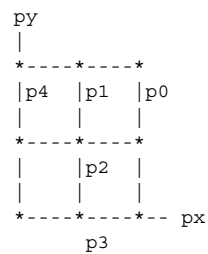
2. Monotonicity Conditions

$px(i) \leq px(i+1)$
 $py(i) \leq py(i+1)$

3. Local Continuity Constraints

See Table 4.5 on page 211 in Rabiner and Juang's book.

Itakura Type:



Allowable paths are

$p1 \rightarrow p0$ (1, 0)
 $p2 \rightarrow p0$ (1, 1)
 $p3 \rightarrow p0$ (1, 2)

Consecutive (1, 0) (1, 0) is disallowed. So path $p4 \rightarrow p1 \rightarrow p0$ is disallowed.

4. Global Path Constraints

Due to local continuity constraints, certain portions of the (px, py) plane are excluded from the region the optimal warping path can transverse. This forms global path constraints.

5. Slope Weighting

See Equation 4.150-3 on page 216 in Rabiner and Juang's book.

A path in (px, py) plane can be represented in chain code. The value of the chain code is defined as following.

```

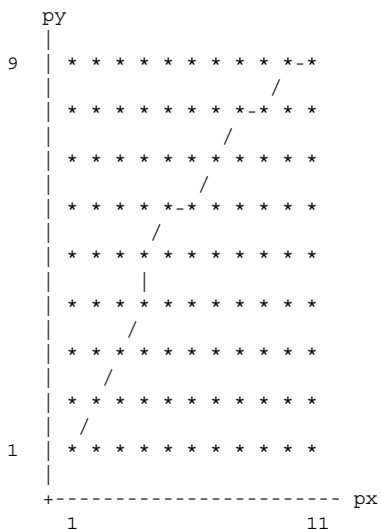
=====
shift ( x , y ) | chain code
-----
( 1 , 0 ) | 0
( 0 , 1 ) | 1
( 1 , 1 ) | 2
( 2 , 1 ) | 3
( 1 , 2 ) | 4
( 3 , 1 ) | 5
( 3 , 2 ) | 6
( 1 , 3 ) | 7
    
```

mllib_SignalDTWScalar_S16(3MLIB)

```
( 2 , 3 ) | 8
=====
py
|
* 8 7 *
|
* 4 * 6
|
1 2 3 5
|
x--0--*--*-- px
```

where x marks the start point of a path segment, the numbers are the values of the chain code for the segment that ends at the point.

In following example, the observed data with 11 data points are mapped into the reference data with 9 data points



The chain code that represents the path is

```
(2 2 2 1 2 0 2 2 0 2 0)
```

See *Fundamentals of Speech Recognition* by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.

PARAMETERS

The function takes the following arguments:

- dist* The distance of the optimal path.
- dobs* The observed data array.
- lobs* The length of the observed data array.
- sobs* The scaling factor of the observed data array, where `actual_data = input_data * 2**(-scaling_factor)`.

mllib_SignalDTWScalar_S16(3MLIB)

state Pointer to the internal state structure.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalDTWScalarInit_S16(3MLIB)`,
`mllib_SignalDTWScalar_S16(3MLIB)`,
`mllib_SignalDTWScalarPath_S16(3MLIB)`,
`mllib_SignalDTWScalarFree_S16(3MLIB)`, `attributes(5)`

mllib_SignalDTWVector_F32(3MLIB)

NAME	mllib_SignalDTWVector_F32 – perform dynamic time warping on vector data
SYNOPSIS	<pre>cc [flag...] file... -lmlib [library...] #include <mllib.h> mllib_status mllib_SignalDTWVector_F32(mllib_d64 *dist, const mllib_f32 **dobs, mllib_s32 lobs, void *state);</pre>
DESCRIPTION	<p>The <code>mllib_SignalDTWVector_F32()</code> function performs dynamic time warping on vector data.</p> <p>Assume the reference data are</p> $r(y), y=1,2,\dots,N$ <p>and the observed data are</p> $o(x), x=1,2,\dots,M$ <p>the dynamic time warping is to find a mapping function (a path)</p> $p(i) = \{px(i), py(i)\}, i=1,2,\dots,Q$ <p>with the minimum distance.</p> <p>In K-best paths case, K paths with the K minimum distances are searched.</p> <p>The distance of a path is defined as</p> $\text{dist} = \sum_{i=1}^Q d(r(py(i)), o(px(i))) * m(px(i), py(i))$ <p>where $d(r, o)$ is the dissimilarity between data point/vector r and data point/vector o; $m(x, y)$ is the path weighting coefficient associated with path point (x, y); N is the length of the reference data; M is the length of the observed data; Q is the length of the path.</p> <p>Using L1 norm (sum of absolute differences)</p> $d(r, o) = \sum_{i=0}^{L-1} r(i) - o(i) $ <p>Using L2 norm (Euclidean distance)</p> $d(r, o) = \text{SQRT} \left\{ \sum_{i=0}^{L-1} (r(i) - o(i))^2 \right\}$ <p>where L is the length of each data vector.</p> <p>To scalar data where $L=1$, the two norms are the same.</p> $d(r, o) = r - o = \text{SQRT} \left\{ (r - o)^2 \right\}$ <p>The constraints of dynamic time warping are:</p>

1. Endpoint constraints

$px(1) = 1$
 $1 \leq py(1) \leq 1 + \text{delta}$
 and

$px(Q) = M$
 $N - \text{delta} \leq py(Q) \leq N$

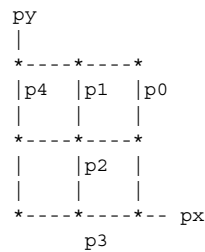
2. Monotonicity Conditions

$px(i) \leq px(i+1)$
 $py(i) \leq py(i+1)$

3. Local Continuity Constraints

See Table 4.5 on page 211 in Rabiner and Juang's book.

Itakura Type:



Allowable paths are

$p1 \rightarrow p0$ (1, 0)
 $p2 \rightarrow p0$ (1, 1)
 $p3 \rightarrow p0$ (1, 2)

Consecutive (1, 0) (1, 0) is disallowed. So path $p4 \rightarrow p1 \rightarrow p0$ is disallowed.

4. Global Path Constraints

Due to local continuity constraints, certain portions of the (px, py) plane are excluded from the region the optimal warping path can transverse. This forms global path constraints.

5. Slope Weighting

See Equation 4.150-3 on page 216 in Rabiner and Juang's book.

A path in (px, py) plane can be represented in chain code. The value of the chain code is defined as following.

```

=====
shift ( x , y ) | chain code
-----
( 1 , 0 ) | 0
( 0 , 1 ) | 1
( 1 , 1 ) | 2
( 2 , 1 ) | 3
( 1 , 2 ) | 4
( 3 , 1 ) | 5
( 3 , 2 ) | 6
( 1 , 3 ) | 7
    
```

mllib_SignalDTWVector_F32(3MLIB)

```

( 2 , 3 ) | 8
=====
py
|
* 8 7 *
|
* 4 * 6
|
1 2 3 5
|
x--0--*--*-- px

```

where x marks the start point of a path segment, the numbers are the values of the chain code for the segment that ends at the point.

In following example, the observed data with 11 data points are mapped into the reference data with 9 data points

```

py
|
9 | * * * * * * * * * * *
| * * * * * * * * * * *
| * * * * * * * * * * *
| * * * * * * * * * * *
| * * * * * * * * * * *
| * * * * * * * * * * *
| * * * * * * * * * * *
| * * * * * * * * * * *
| * * * * * * * * * * *
| * * * * * * * * * * *
| * * * * * * * * * * *
1 | * * * * * * * * * * *
+-----+----- px
1 11

```

The chain code that represents the path is

```
( 2 2 2 1 2 0 2 2 0 2 0 )
```

See *Fundamentals of Speech Recognition* by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.

PARAMETERS

The function takes the following arguments:

- dist* The distance of the optimal path.
- dobs* The observed data array.
- lobs* The length of the observed data array.
- state* Pointer to the internal state structure.

mllib_SignalDTWVector_F32(3MLIB)

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalDTWVectorInit_F32(3MLIB)`,
`mllib_SignalDTWVector_F32(3MLIB)`,
`mllib_SignalDTWVectorPath_F32(3MLIB)`,
`mllib_SignalDTWVectorFree_F32(3MLIB)`, `attributes(5)`

mllib_SignalDTWVectorFree_S16(3MLIB)

NAME mllib_SignalDTWVectorFree_S16, mllib_SignalDTWVectorFree_F32 – clean up for vector data

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_SignalDTWVectorFree_S16(void *state);
void mllib_SignalDTWVectorFree_F32(void *state);
```

DESCRIPTION Each of these functions frees the internal state structure for dynamic time warping (DTW) of vector data.

This function cleans up the internal state structure and releases all memory buffers.

PARAMETERS Each of the functions takes the following arguments:

state Pointer to the internal state structure.

RETURN VALUES None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalDTWVectorInit_S16(3MLIB),
mllib_SignalDTWVectorInit_F32(3MLIB),
mllib_SignalDTWVector_S16(3MLIB), mllib_SignalDTWVector_F32(3MLIB),
mllib_SignalDTWVectorPath_S16(3MLIB),
mllib_SignalDTWVectorPath_F32(3MLIB), attributes(5)

NAME mllib_SignalDTWVectorInit_F32 – initialization for vector data

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalDTWVectorInit_F32(void *state, const
      mllib_f32 **dref, mllib_s32 lref, mllib_s32 ndata, mllib_s32 dtype,
      mllib_s32 delta, mllib_s32 local, mllib_s32 slope);
```

DESCRIPTION The mllib_SignalDTWVectorInit_F32() function initializes the internal state structure for dynamic time warping (DTW) of vector data.

The init function performs internal state structure allocation and global initialization. Per DTW function call initialization is done in DTW function, so the same internal state structure can be reused for multiple DTW function calls.

PARAMETERS The function takes the following arguments:

dref The reference data array.

lref The length of the reference data array.

ndata The length of each data vector.

dtype The type of distance metric between data vectors. MLIB_DTW_L1NORM for L1 norm of difference (sum of absolute difference). MLIB_DTW_L2NORM for L2 norm of difference (Euclidean distance).

delta The delta in the endpoint constraints.

local The type of the local continuity constraints. MLIB_DTW_ITAKURA for Itakura type constraints.

slope The type of the slope weighting. MLIB_DTW_NONE for no slope weighting.

state Pointer to the internal state structure.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalDTWVectorInit_F32(3MLIB), mllib_SignalDTWVector_F32(3MLIB), mllib_SignalDTWVectorPath_F32(3MLIB), mllib_SignalDTWVectorFree_F32(3MLIB), attributes(5)

mllib_SignalDTWVectorInit_S16(3MLIB)

NAME	mllib_SignalDTWVectorInit_S16 – initialization for vector data						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalDTWVectorInit_S16(void *state, const mllib_s16 **dref, mllib_s32 lref, mllib_s32 ndata, mllib_s32 dtype, mllib_s32 sref, mllib_s32 delta, mllib_s32 local, mllib_s32 slope);</pre>						
DESCRIPTION	<p>The <code>mllib_SignalDTWVectorInit_S16()</code> function initializes the internal state structure for dynamic time warping (DTW) of vector data.</p> <p>The <code>init</code> function performs internal state structure allocation and global initialization. Per DTW function call initialization is done in DTW function, so the same internal state structure can be reused for multiple DTW function calls.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dref</i> The reference data array.</p> <p><i>lref</i> The length of the reference data array.</p> <p><i>ndata</i> The length of each data vector.</p> <p><i>dtype</i> The type of distance metric between data vectors. MLIB_DTW_L1NORM for L1 norm of difference (sum of absolute difference). MLIB_DTW_L2NORM for L2 norm of difference (Euclidean distance).</p> <p><i>sref</i> The scaling factor of the reference data array, where <code>actual_data = input_data * 2**(-scaling_factor)</code>.</p> <p><i>delta</i> The delta in the endpoint constraints.</p> <p><i>local</i> The type of the local continuity constraints. MLIB_DTW_ITAKURA for Itakura type constraints.</p> <p><i>slope</i> The type of the slope weighting. MLIB_DTW_NONE for no slope weighting.</p> <p><i>state</i> Pointer to the internal state structure.</p>						
RETURN VALUES	The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"> <thead> <tr> <th>ATTRIBUTE TYPE</th> <th>ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						

`mllib_SignalDTWVectorInit_S16(3MLIB)`

SEE ALSO `mllib_SignalDTWVectorInit_S16(3MLIB)`,
`mllib_SignalDTWVector_S16(3MLIB)`,
`mllib_SignalDTWVectorPath_S16(3MLIB)`,
`mllib_SignalDTWVectorFree_S16(3MLIB)`, `attributes(5)`

mllib_SignalDTWVectorPath_F32(3MLIB)

NAME	mllib_SignalDTWVectorPath_F32 – perform dynamic time warping on vector data
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalDTWVectorPath_F32(mllib_d64 *dist, mllib_s32 *path, mllib_s32 *lpath, const mllib_f32 **dobs, mllib_s32 lobs, void *state);</pre>
DESCRIPTION	<p>The mllib_SignalDTWVectorPath_F32() function performs dynamic time warping on vector data.</p> <p>Assume the reference data are</p> $r(y), y=1,2,\dots,N$ <p>and the observed data are</p> $o(x), x=1,2,\dots,M$ <p>the dynamic time warping is to find a mapping function (a path)</p> $p(i) = \{px(i), py(i)\}, i=1,2,\dots,Q$ <p>with the minimum distance.</p> <p>In K-best paths case, K paths with the K minimum distances are searched.</p> <p>The distance of a path is defined as</p> $\text{dist} = \sum_{i=1}^Q d(r(py(i)), o(px(i))) * m(px(i), py(i))$ <p>where $d(r, o)$ is the dissimilarity between data point/vector r and data point/vector o; $m(x, y)$ is the path weighting coefficient associated with path point (x, y); N is the length of the reference data; M is the length of the observed data; Q is the length of the path.</p> <p>Using L1 norm (sum of absolute differences)</p> $d(r, o) = \sum_{i=0}^{L-1} r(i) - o(i) $ <p>Using L2 norm (Euclidean distance)</p> $d(r, o) = \text{SQRT} \left\{ \sum_{i=0}^{L-1} (r(i) - o(i))^2 \right\}$ <p>where L is the length of each data vector.</p> <p>To scalar data where $L=1$, the two norms are the same.</p> $d(r, o) = r - o = \text{SQRT} \left\{ (r - o)^2 \right\}$ <p>The constraints of dynamic time warping are:</p>

1. Endpoint constraints

$px(1) = 1$
 $1 \leq py(1) \leq 1 + \text{delta}$
 and

$px(Q) = M$
 $N - \text{delta} \leq py(Q) \leq N$

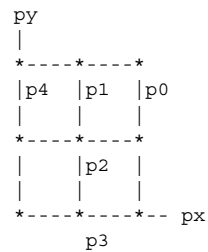
2. Monotonicity Conditions

$px(i) \leq px(i+1)$
 $py(i) \leq py(i+1)$

3. Local Continuity Constraints

See Table 4.5 on page 211 in Rabiner and Juang's book.

Itakura Type:



Allowable paths are

$p1 \rightarrow p0$ (1, 0)
 $p2 \rightarrow p0$ (1, 1)
 $p3 \rightarrow p0$ (1, 2)

Consecutive (1, 0) (1, 0) is disallowed. So path $p4 \rightarrow p1 \rightarrow p0$ is disallowed.

4. Global Path Constraints

Due to local continuity constraints, certain portions of the (px, py) plane are excluded from the region the optimal warping path can transverse. This forms global path constraints.

5. Slope Weighting

See Equation 4.150-3 on page 216 in Rabiner and Juang's book.

A path in (px, py) plane can be represented in chain code. The value of the chain code is defined as following.

```

=====
shift ( x , y ) | chain code
-----
( 1 , 0 ) | 0
( 0 , 1 ) | 1
( 1 , 1 ) | 2
( 2 , 1 ) | 3
( 1 , 2 ) | 4
( 3 , 1 ) | 5
( 3 , 2 ) | 6
( 1 , 3 ) | 7
    
```

mllib_SignalDTWVectorPath_F32(3MLIB)

```

( 2 , 3 ) | 8
=====
py
|
* 8 7 *
|
* 4 * 6
|
1 2 3 5
|
x--0--*--*-- px

```

where x marks the start point of a path segment, the numbers are the values of the chain code for the segment that ends at the point.

In following example, the observed data with 11 data points are mapped into the reference data with 9 data points

```

py
|
9 | * * * * * * * * * *
| * * * * * * * * * *
| * * * * * * * * * *
| * * * * * * * * * *
| * * * * * * * * * *
| * * * * * * * * * *
| * * * * * * * * * *
| * * * * * * * * * *
| * * * * * * * * * *
| * * * * * * * * * *
| * * * * * * * * * *
| * * * * * * * * * *
1 | * * * * * * * * * *
|
+----- px
1 11

```

The chain code that represents the path is

```
(2 2 2 1 2 0 2 2 0 2 0)
```

See *Fundamentals of Speech Recognition* by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.

PARAMETERS

The function takes the following arguments:

- dist* The distance of the optimal path.
- path* The optimal path.
- lpath* The length of the optimal path.
- dobs* The observed data array.

mllib_SignalDTWVectorPath_F32(3MLIB)

lobs The length of the observed data array.

state Pointer to the internal state structure.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalDTWVectorInit_F32(3MLIB)`,
`mllib_SignalDTWVector_F32(3MLIB)`,
`mllib_SignalDTWVectorPath_F32(3MLIB)`,
`mllib_SignalDTWVectorFree_F32(3MLIB)`, `attributes(5)`

mllib_SignalDTWVectorPath_S16(3MLIB)

NAME	mllib_SignalDTWVectorPath_S16 – perform dynamic time warping on vector data
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalDTWVectorPath_S16(mllib_d64 *dist, mllib_s32 *path, mllib_s32 *lpath, const mllib_s16 **dobs, mllib_s32 lobs, mllib_s32 sob, void *state);</pre>
DESCRIPTION	<p>The mllib_SignalDTWVectorPath_S16() function performs dynamic time warping on vector data.</p> <p>Assume the reference data are</p> $r(y), y=1,2,\dots,N$ <p>and the observed data are</p> $o(x), x=1,2,\dots,M$ <p>the dynamic time warping is to find a mapping function (a path)</p> $p(i) = \{px(i), py(i)\}, i=1,2,\dots,Q$ <p>with the minimum distance.</p> <p>In K-best paths case, K paths with the K minimum distances are searched.</p> <p>The distance of a path is defined as</p> $\text{dist} = \sum_{i=1}^Q d(r(py(i)), o(px(i))) * m(px(i), py(i))$ <p>where $d(r, o)$ is the dissimilarity between data point/vector r and data point/vector o; $m(x, y)$ is the path weighting coefficient associated with path point (x, y); N is the length of the reference data; M is the length of the observed data; Q is the length of the path.</p> <p>Using L1 norm (sum of absolute differences)</p> $d(r, o) = \sum_{i=0}^{L-1} r(i) - o(i) $ <p>Using L2 norm (Euclidean distance)</p> $d(r, o) = \text{SQRT} \left\{ \sum_{i=0}^{L-1} (r(i) - o(i))^2 \right\}$ <p>where L is the length of each data vector.</p> <p>To scalar data where $L=1$, the two norms are the same.</p> $d(r, o) = r - o = \text{SQRT} \left\{ (r - o)^2 \right\}$ <p>The constraints of dynamic time warping are:</p>

1. Endpoint constraints

$px(1) = 1$
 $1 \leq py(1) \leq 1 + \text{delta}$
 and

$px(Q) = M$
 $N - \text{delta} \leq py(Q) \leq N$

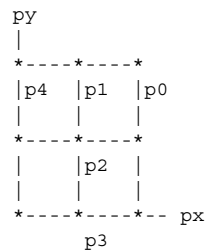
2. Monotonicity Conditions

$px(i) \leq px(i+1)$
 $py(i) \leq py(i+1)$

3. Local Continuity Constraints

See Table 4.5 on page 211 in Rabiner and Juang's book.

Itakura Type:



Allowable paths are

$p1 \rightarrow p0$ (1, 0)
 $p2 \rightarrow p0$ (1, 1)
 $p3 \rightarrow p0$ (1, 2)

Consecutive (1, 0) (1, 0) is disallowed. So path $p4 \rightarrow p1 \rightarrow p0$ is disallowed.

4. Global Path Constraints

Due to local continuity constraints, certain portions of the (px, py) plane are excluded from the region the optimal warping path can transverse. This forms global path constraints.

5. Slope Weighting

See Equation 4.150-3 on page 216 in Rabiner and Juang's book.

A path in (px, py) plane can be represented in chain code. The value of the chain code is defined as following.

```

=====
shift ( x , y ) | chain code
-----
( 1 , 0 ) | 0
( 0 , 1 ) | 1
( 1 , 1 ) | 2
( 2 , 1 ) | 3
( 1 , 2 ) | 4
( 3 , 1 ) | 5
( 3 , 2 ) | 6
( 1 , 3 ) | 7
    
```

mllib_SignalDTWVectorPath_S16(3MLIB)

```

      ( 2 , 3 ) |      8
      =====
      py
      |
      * 8 7 *
      |
      * 4 * 6
      |
      1 2 3 5
      |
      x--0--*--*-- px
  
```

where x marks the start point of a path segment, the numbers are the values of the chain code for the segment that ends at the point.

In following example, the observed data with 11 data points are mapped into the reference data with 9 data points

```

      py
      |
      9 | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      | * * * * * * * * * *
      1 | * * * * * * * * * *
      |
      +----- px
      1                               11
  
```

The chain code that represents the path is

```
( 2 2 2 1 2 0 2 2 0 2 0 )
```

See *Fundamentals of Speech Recognition* by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.

PARAMETERS

The function takes the following arguments:

- dist* The distance of the optimal path.
- path* The optimal path.
- lpath* The length of the optimal path.
- dobs* The observed data array.

`mllib_SignalDTWVectorPath_S16(3MLIB)`

lobs The length of the observed data array.
sobs The scaling factor of the observed data array, where `actual_data = input_data * 2**(-scaling_factor)`.
state Pointer to the internal state structure.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalDTWVectorInit_S16(3MLIB)`,
`mllib_SignalDTWVector_S16(3MLIB)`,
`mllib_SignalDTWVectorPath_S16(3MLIB)`,
`mllib_SignalDTWVectorFree_S16(3MLIB)`, `attributes(5)`

mllib_SignalDTWVector_S16(3MLIB)

NAME	mllib_SignalDTWVector_S16 – perform dynamic time warping on vector data
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalDTWVector_S16(mllib_d64 *dist, const mllib_s16 **dobs, mllib_s32 lobs, mllib_s32 sob, void *state);</pre>
DESCRIPTION	<p>The <code>mllib_SignalDTWVector_S16()</code> function performs dynamic time warping on vector data.</p> <p>Assume the reference data are</p> $r(y), y=1,2,\dots,N$ <p>and the observed data are</p> $o(x), x=1,2,\dots,M$ <p>the dynamic time warping is to find a mapping function (a path)</p> $p(i) = \{px(i),py(i)\}, i=1,2,\dots,Q$ <p>with the minimum distance.</p> <p>In K-best paths case, K paths with the K minimum distances are searched.</p> <p>The distance of a path is defined as</p> $\text{dist} = \sum_{i=1}^Q d(r(py(i)),o(px(i))) * m(px(i),py(i))$ <p>where $d(r, o)$ is the dissimilarity between data point/vector r and data point/vector o; $m(x, y)$ is the path weighting coefficient associated with path point (x, y); N is the length of the reference data; M is the length of the observed data; Q is the length of the path.</p> <p>Using L1 norm (sum of absolute differences)</p> $d(r,o) = \sum_{i=0}^{L-1} r(i) - o(i) $ <p>Using L2 norm (Euclidean distance)</p> $d(r,o) = \text{SQRT} \{ \sum_{i=0}^{L-1} (r(i) - o(i))^{**2} \}$ <p>where L is the length of each data vector.</p> <p>To scalar data where $L=1$, the two norms are the same.</p> $d(r,o) = r - o = \text{SQRT} \{ (r - o)^{**2} \}$ <p>The constraints of dynamic time warping are:</p>

1. Endpoint constraints

$px(1) = 1$
 $1 \leq py(1) \leq 1 + \text{delta}$
 and

$px(Q) = M$
 $N - \text{delta} \leq py(Q) \leq N$

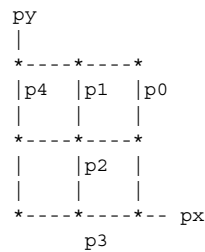
2. Monotonicity Conditions

$px(i) \leq px(i+1)$
 $py(i) \leq py(i+1)$

3. Local Continuity Constraints

See Table 4.5 on page 211 in Rabiner and Juang's book.

Itakura Type:



Allowable paths are

$p1 \rightarrow p0$ (1, 0)
 $p2 \rightarrow p0$ (1, 1)
 $p3 \rightarrow p0$ (1, 2)

Consecutive (1, 0) (1, 0) is disallowed. So path $p4 \rightarrow p1 \rightarrow p0$ is disallowed.

4. Global Path Constraints

Due to local continuity constraints, certain portions of the (px, py) plane are excluded from the region the optimal warping path can transverse. This forms global path constraints.

5. Slope Weighting

See Equation 4.150-3 on page 216 in Rabiner and Juang's book.

A path in (px, py) plane can be represented in chain code. The value of the chain code is defined as following.

```

=====
shift ( x , y ) | chain code
-----
( 1 , 0 ) | 0
( 0 , 1 ) | 1
( 1 , 1 ) | 2
( 2 , 1 ) | 3
( 1 , 2 ) | 4
( 3 , 1 ) | 5
( 3 , 2 ) | 6
( 1 , 3 ) | 7
    
```

mllib_SignalDTWVector_S16(3MLIB)

```

      ( 2 , 3 ) |      8
      =====
      py
      |
      * 8 7 *
      |
      * 4 * 6
      |
      1 2 3 5
      |
      x--0--*--*-- px
  
```

where x marks the start point of a path segment, the numbers are the values of the chain code for the segment that ends at the point.

In following example, the observed data with 11 data points are mapped into the reference data with 9 data points

```

      py
      |
      9 * * * * * * * * * *
      | * * * * * * * * * * /
      | * * * * * * * * * * /
      | * * * * * * * * * * /
      | * * * * * * * * * * /
      | * * * * * * * * * * /
      | * * * * * * * * * * /
      | * * * * * * * * * * /
      | * * * * * * * * * * /
      | * * * * * * * * * * /
      | * * * * * * * * * * /
      | * * * * * * * * * * /
      | * * * * * * * * * * /
      | * * * * * * * * * * /
      | * * * * * * * * * * /
      | * * * * * * * * * * /
      | * * * * * * * * * * /
      1 * * * * * * * * * *
      |
      +----- px
      1                          11
  
```

The chain code that represents the path is

```
( 2 2 2 1 2 0 2 2 0 2 0 )
```

See *Fundamentals of Speech Recognition* by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.

PARAMETERS

The function takes the following arguments:

<i>dist</i>	The distance of the optimal path.
<i>dobs</i>	The observed data array.
<i>lobs</i>	The length of the observed data array.
<i>sobs</i>	The scaling factor of the observed data array, where <code>actual_data = input_data * 2**(-scaling_factor)</code> .

`mllib_SignalDTWVector_S16(3MLIB)`

state Pointer to the internal state structure.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalDTWVectorInit_S16(3MLIB)`,
`mllib_SignalDTWVector_S16(3MLIB)`,
`mllib_SignalDTWVectorPath_S16(3MLIB)`,
`mllib_SignalDTWVectorFree_S16(3MLIB)`, `attributes(5)`

mllib_SignalEmphasizeFree_S16_S16(3MLIB)

NAME mllib_SignalEmphasizeFree_S16_S16, mllib_SignalEmphasizeFree_S16S_S16S, mllib_SignalEmphasizeFree_F32_F32, mllib_SignalEmphasizeFree_F32S_F32S – clean up for signal pre-emphasizing

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_SignalEmphasizeFree_S16_S16(void *filter) ;
void mllib_SignalEmphasizeFree_S16S_S16S(void *filter) ;
void mllib_SignalEmphasizeFree_F32_F32(void *filter) ;
void mllib_SignalEmphasizeFree_F32S_F32S(void *filter) ;
```

DESCRIPTION Each of these functions releases the memory allocated for the internal state's structure.

PARAMETERS Each of the functions takes the following arguments:
filter Internal filter structure.

RETURN VALUES None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalEmphasize_S16_S16_Sat(3MLIB), mllib_SignalEmphasizeInit_S16_S16(3MLIB), attributes(5)

mlib_SignalEmphasizeInit_S16_S16(3MLIB)

NAME mlib_SignalEmphasizeInit_S16_S16, mlib_SignalEmphasizeInit_S16S_S16S, mlib_SignalEmphasizeInit_F32_F32, mlib_SignalEmphasizeInit_F32S_F32S – initialization for signal pre-emphasizing

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_SignalEmphasizeInit_S16_S16(void **filter,
      mlib_f32 alpha);

mlib_status mlib_SignalEmphasizeInit_S16S_S16S(void **filter,
      mlib_f32 alpha);

mlib_status mlib_SignalEmphasizeInit_F32_F32(void **filter,
      mlib_f32 alpha);

mlib_status mlib_SignalEmphasizeInit_F32S_F32S(void **filter,
      mlib_f32 alpha);
```

DESCRIPTION Each of these functions allocates memory for an internal filter structure and converts the filter coefficients into the internal representation.

PARAMETERS Each of the functions takes the following arguments:

filter Internal filter structure.

alpha Emphasizing coefficient. 0 < alpha < 1.0

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mlib_SignalEmphasize_S16_S16_Sat(3MLIB), mlib_SignalEmphasizeFree_S16_S16(3MLIB), attributes(5)

mllib_SignalEmphasize_S16_S16_Sat(3MLIB)

NAME	mllib_SignalEmphasize_S16_S16_Sat, mllib_SignalEmphasize_S16S_S16S_Sat, mllib_SignalEmphasize_F32_F32, mllib_SignalEmphasize_F32S_F32S – signal pre-emphasizing				
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalEmphasize_S16_S16_Sat(mllib_s16 *dst, const mllib_s16 *src, void *filter, mllib_s32 n); mllib_status mllib_SignalEmphasize_S16S_S16S_Sat(mllib_s16 *dst, const mllib_s16 *src, void *filter, mllib_s32 n); mllib_status mllib_SignalEmphasize_F32_F32(mllib_f32 *dst, const mllib_f32 *src, void *filter, mllib_s32 n); mllib_status mllib_SignalEmphasize_F32S_F32S(mllib_f32 *dst, const mllib_f32 *src, void *filter, mllib_s32 n);</pre>				
DESCRIPTION	<p>Each of these functions applies the preemphasizer to one signal packet and updates the filter states.</p> <p>For monaural signals, the following equation is used:</p> $dst[i] = src[i] - \alpha * src[i - 1]$ <p>where $i = 0, 1, \dots, (n - 1)$; $src[-1] = 0$.</p> <p>For stereo signals, the following equation is used:</p> $\begin{aligned} dst[2*i] &= src[2*i] - \alpha * src[2*(i - 1)] \\ dst[2*i + 1] &= src[2*i + 1] - \alpha * src[2*(i - 1) + 1] \end{aligned}$ <p>where $i = 0, 1, \dots, (n - 1)$; $src[-2] = src[-1] = 0$.</p>				
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <p><i>dst</i> Destination signal array.</p> <p><i>src</i> Source signal array.</p> <p><i>filter</i> Internal filter structure.</p> <p><i>n</i> Number of samples in the source signal array.</p>				
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.				
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:				
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Interface Stability</td> <td style="text-align: center;">Evolving</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving
ATTRIBUTE TYPE	ATTRIBUTE VALUE				
Interface Stability	Evolving				

`mllib_SignalEmphasize_S16_S16_Sat(3MLIB)`

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	MT-Safe

SEE ALSO `mllib_SignalEmphasizeFree_S16_S16(3MLIB)`,
`mllib_SignalEmphasizeInit_S16_S16(3MLIB)`, `attributes(5)`

mllib_SignalFFT_1(3MLIB)

NAME	mllib_SignalFFT_1, mllib_SignalFFT_1_S16_S16_Mod, mllib_SignalFFT_1_S16C_S16C_Mod, mllib_SignalFFT_1_S16C_S16_Mod, mllib_SignalFFT_1_S16_Mod, mllib_SignalFFT_1_S16C_Mod, mllib_SignalFFT_1_F32_F32, mllib_SignalFFT_1_F32C_F32C, mllib_SignalFFT_1_F32C_F32, mllib_SignalFFT_1_F32, mllib_SignalFFT_1_F32C, mllib_SignalFFT_1_D64_D64, mllib_SignalFFT_1_D64C_D64C, mllib_SignalFFT_1_D64C_D64, mllib_SignalFFT_1_D64, mllib_SignalFFT_1_D64C – signal Fast Fourier Transform (FFT)
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalFFT_1_S16_S16_Mod(mllib_s16 *dstr, mllib_s16 *dsti, const mllib_s16 *srcr, const mllib_s16 *srci, mllib_s32 order); mllib_status mllib_SignalFFT_1_S16C_S16C_Mod(mllib_s16 *dstc, const mllib_s16 *srcc, mllib_s32 order); mllib_status mllib_SignalFFT_1_S16C_S16_Mod(mllib_s16 *dstc, const mllib_s16 *srcr, mllib_s32 order); mllib_status mllib_SignalFFT_1_S16_Mod(mllib_s16 *srcdstr, mllib_s16 *srcdsti, mllib_s32 order); mllib_status mllib_SignalFFT_1_S16C_Mod(mllib_s16 *srcdstc, mllib_s32 order); mllib_status mllib_SignalFFT_1_F32_F32(mllib_f32 *dstr, mllib_f32 *dsti, const mllib_f32 *srcr, const mllib_f32 *srci, mllib_s32 order); mllib_status mllib_SignalFFT_1_F32C_F32C(mllib_f32 *dstc, const mllib_f32 *srcc, mllib_s32 order); mllib_status mllib_SignalFFT_1_F32C_F32(mllib_f32 *dstc, const mllib_f32 *srcr, mllib_s32 order); mllib_status mllib_SignalFFT_1_F32(mllib_f32 *srcdstr, mllib_f32 *srcdsti, mllib_s32 order); mllib_status mllib_SignalFFT_1_F32C(mllib_f32 *srcdstc, mllib_s32 order); mllib_status mllib_SignalFFT_1_D64_D64(mllib_d64 *dstr, mllib_d64 *dsti, const mllib_d64 *srcr, const mllib_d64 *srci, mllib_s32 order); mllib_status mllib_SignalFFT_1_D64C_D64C(mllib_d64 *dstc, const mllib_d64 *srcc, mllib_s32 order); mllib_status mllib_SignalFFT_1_D64C_D64(mllib_d64 *dstc, const mllib_d64 *srcr, mllib_s32 order); mllib_status mllib_SignalFFT_1_D64(mllib_d64 *srcdstr, mllib_d64 *srcdsti, mllib_s32 order); mllib_status mllib_SignalFFT_1_D64C(mllib_d64 *srcdstc, mllib_s32 order);</pre>

DESCRIPTION

Each of the functions in this group performs Fast Fourier Transform (FFT).

The following equation is used for forward FFT:

$$\text{dst}[k] = \frac{1}{C1} \sum_{n=0}^{N-1} \{\text{src}[n] * \exp(-j2*PI*n*k/N)\}$$

and the following equation is used for inverse FFT (IFFT):

$$\text{dst}[n] = \frac{1}{C2} \sum_{k=0}^{N-1} \{\text{src}[k] * \exp(j2*PI*n*k/N)\}$$

where

$$\begin{aligned} k &= 0, 1, \dots, (N - 1) \\ n &= 0, 1, \dots, (N - 1) \\ N &= 2**\text{order} \end{aligned}$$

The signal FFT/IFFT functions can be categorized into four groups according to the ScaleMode in the function names in the following form:

```
mllib_Signal [FFT|IFFT]_ScaleMode_OutType_InType_OpMode ()
mllib_Signal [FFT|IFFT]_ScaleMode_DataType_OpMode ()
```

The scaling factors C1 and C2 used in the equations are defined as follows:

- For ScaleMode = 1, C1 = 1 and C2 = 2**order.
- For ScaleMode = 2, C1 = 2**order and C2 = 1.
- For ScaleMode = 3, C1 = C2 = 2** (order/2) when order is even, or C1 = 2** ((order+1)/2) and C2 = 2** ((order-1)/2) when order is odd.
- For ScaleMode = 4, C1 = 2**P and C2 = 2**Q, where P and Q are adaptive scaling factors and are generated by the functions.

For functions with only real parts for the source signal, the imaginary parts are assumed to be all zero. For functions with only real parts for the destination signal, the imaginary parts are discarded. The functions with only one data type in their names perform the operation in place.

PARAMETERS

Each function takes some of the following arguments:

<i>dst</i>	Destination signal array that contains the real parts.
<i>dsti</i>	Destination signal array that contains the imaginary parts.
<i>srcr</i>	Source signal array that contains the real parts.
<i>srci</i>	Source signal array that contains the imaginary parts.
<i>dstc</i>	Complex destination signal array. <i>dstc</i> [2*i] contains the real parts, and <i>dstc</i> [2*i+1] contains the imaginary parts.
<i>src</i>	Complex source signal array. <i>src</i> [2*i] contains the real parts, and <i>src</i> [2*i+1] contains the imaginary parts.

mllib_SignalFFT_1(3MLIB)

<i>srcdstr</i>	Source and destination signal array that contains the real parts.
<i>srcdsti</i>	Source and destination signal array that contains the imaginary parts.
<i>srcdstc</i>	Complex source and destination signal array. <i>srcdstc</i> [2*i] contains the real parts, and <i>srcdstc</i> [2*i+1] contains the imaginary parts.
<i>order</i>	Order of the transformation. The base-2 logarithm of the number of data samples.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalFFT_2(3MLIB)`, `mllib_SignalFFT_3(3MLIB)`, `mllib_SignalFFT_4(3MLIB)`, `mllib_SignalIFFT_1(3MLIB)`, `mllib_SignalIFFT_2(3MLIB)`, `mllib_SignalIFFT_3(3MLIB)`, `mllib_SignalIFFT_4(3MLIB)`, `attributes(5)`

NAME	mllib_SignalFFT_2, mllib_SignalFFT_2_S16_S16, mllib_SignalFFT_2_S16C_S16C, mllib_SignalFFT_2_S16C_S16, mllib_SignalFFT_2_S16, mllib_SignalFFT_2_S16C, mllib_SignalFFT_2_F32_F32, mllib_SignalFFT_2_F32C_F32C, mllib_SignalFFT_2_F32C_F32, mllib_SignalFFT_2_F32, mllib_SignalFFT_2_F32C, mllib_SignalFFT_2_D64_D64, mllib_SignalFFT_2_D64C_D64C, mllib_SignalFFT_2_D64C_D64, mllib_SignalFFT_2_D64, mllib_SignalFFT_2_D64C – signal Fast Fourier Transform (FFT)
SYNOPSIS	<pre>cc [flag...] file... -lmlib [library...] #include <mllib.h> mllib_status mllib_SignalFFT_2_S16_S16(mllib_s16 *dst, mllib_s16 *dsti, const mllib_s16 *srcr, const mllib_s16 *srci, mllib_s32 order); mllib_status mllib_SignalFFT_2_S16C_S16C(mllib_s16 *dstc, const mllib_s16 *srcc, mllib_s32 order); mllib_status mllib_SignalFFT_2_S16C_S16(mllib_s16 *dstc, const mllib_s16 *srcr, mllib_s32 order); mllib_status mllib_SignalFFT_2_S16(mllib_s16 *srcdst, mllib_s16 *srcdsti, mllib_s32 order); mllib_status mllib_SignalFFT_2_S16C(mllib_s16 *srcdstc, mllib_s32 order); mllib_status mllib_SignalFFT_2_F32_F32(mllib_f32 *dst, mllib_f32 *dsti, const mllib_f32 *srcr, const mllib_f32 *srci, mllib_s32 order); mllib_status mllib_SignalFFT_2_F32C_F32C(mllib_f32 *dstc, const mllib_f32 *srcc, mllib_s32 order); mllib_status mllib_SignalFFT_2_F32C_F32(mllib_f32 *dstc, const mllib_f32 *srcr, mllib_s32 order); mllib_status mllib_SignalFFT_2_F32(mllib_f32 *srcdst, mllib_f32 *srcdsti, mllib_s32 order); mllib_status mllib_SignalFFT_2_F32C(mllib_f32 *srcdstc, mllib_s32 order); mllib_status mllib_SignalFFT_2_D64_D64(mllib_d64 *dst, mllib_d64 *dsti, const mllib_d64 *srcr, const mllib_d64 *srci, mllib_s32 order); mllib_status mllib_SignalFFT_2_D64C_D64C(mllib_d64 *dstc, const mllib_d64 *srcc, mllib_s32 order); mllib_status mllib_SignalFFT_2_D64C_D64(mllib_d64 *dstc, const mllib_d64 *srcr, mllib_s32 order); mllib_status mllib_SignalFFT_2_D64(mllib_d64 *srcdst, mllib_d64 *srcdsti, mllib_s32 order); mllib_status mllib_SignalFFT_2_D64C(mllib_d64 *srcdstc, mllib_s32 order);</pre>
DESCRIPTION	Each of the functions in this group performs Fast Fourier Transform (FFT).

mllib_SignalFFT_2(3MLIB)

The following equation is used for forward FFT:

$$\text{dst}[k] = \frac{1}{C1} \sum_{n=0}^{N-1} \{\text{src}[n] * \exp(-j2*PI*n*k/N)\}$$

and the following equation is used for inverse FFT (IFFT):

$$\text{dst}[n] = \frac{1}{C2} \sum_{k=0}^{N-1} \{\text{src}[k] * \exp(j2*PI*n*k/N)\}$$

where

$k = 0, 1, \dots, (N - 1)$
 $n = 0, 1, \dots, (N - 1)$
 $N = 2**\text{order}$

The signal FFT/IFFT functions can be categorized into four groups according to the ScaleMode in the function names in the following form:

```
mllib_Signal[FFT|IFFT]_ScaleMode_OutType_InType_OpMode()  
mllib_Signal[FFT|IFFT]_ScaleMode_DataType_OpMode()
```

The scaling factors C1 and C2 used in the equations are defined as follows:

- For ScaleMode = 1, C1 = 1 and C2 = 2**order.
- For ScaleMode = 2, C1 = 2**order and C2 = 1.
- For ScaleMode = 3, C1 = C2 = 2** (order/2) when order is even, or C1 = 2** ((order+1)/2) and C2 = 2** ((order-1)/2) when order is odd.
- For ScaleMode = 4, C1 = 2**P and C2 = 2**Q, where P and Q are adaptive scaling factors and are generated by the functions.

For functions with only real parts for the source signal, the imaginary parts are assumed to be all zero. For functions with only real parts for the destination signal, the imaginary parts are discarded. The functions with only one data type in their names perform the operation in place.

PARAMETERS Each function takes some of the following arguments:

<i>dstr</i>	Destination signal array that contains the real parts.
<i>dsti</i>	Destination signal array that contains the imaginary parts.
<i>srcr</i>	Source signal array that contains the real parts.
<i>srci</i>	Source signal array that contains the imaginary parts.
<i>dstc</i>	Complex destination signal array. <i>dstc</i> [2*i] contains the real parts, and <i>dstc</i> [2*i+1] contains the imaginary parts.
<i>src</i>	Complex source signal array. <i>src</i> [2*i] contains the real parts, and <i>src</i> [2*i+1] contains the imaginary parts.
<i>srcdstr</i>	Source and destination signal array that contains the real parts.

mllib_SignalFFT_2(3MLIB)

srcdsti Source and destination signal array that contains the imaginary parts.

srcdstc Complex source and destination signal array. *srcdstc*[2*i] contains the real parts, and *srcdstc*[2*i+1] contains the imaginary parts.

order Order of the transformation. The base-2 logarithm of the number of data samples.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalFFT_1(3MLIB)`, `mllib_SignalFFT_3(3MLIB)`, `mllib_SignalFFT_4(3MLIB)`, `mllib_SignalIFFT_1(3MLIB)`, `mllib_SignalIFFT_2(3MLIB)`, `mllib_SignalIFFT_3(3MLIB)`, `mllib_SignalIFFT_4(3MLIB)`, `attributes(5)`

mllib_SignalFFT_3(3MLIB)

NAME	mllib_SignalFFT_3, mllib_SignalFFT_3_S16_S16_Mod, mllib_SignalFFT_3_S16C_S16C_Mod, mllib_SignalFFT_3_S16C_S16C_Mod, mllib_SignalFFT_3_S16_Mod, mllib_SignalFFT_3_S16C_Mod, mllib_SignalFFT_3_F32_F32, mllib_SignalFFT_3_F32C_F32C, mllib_SignalFFT_3_F32C_F32, mllib_SignalFFT_3_F32, mllib_SignalFFT_3_F32C, mllib_SignalFFT_3_D64_D64, mllib_SignalFFT_3_D64C_D64C, mllib_SignalFFT_3_D64C_D64, mllib_SignalFFT_3_D64, mllib_SignalFFT_3_D64C – signal Fast Fourier Transform (FFT)
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalFFT_3_S16_S16_Mod(mllib_s16 *dstr, mllib_s16 *dsti, const mllib_s16 *srcr, const mllib_s16 *srci, mllib_s32 order); mllib_status mllib_SignalFFT_3_S16C_S16C_Mod(mllib_s16 *dstc, const mllib_s16 *srcc, mllib_s32 order); mllib_status mllib_SignalFFT_3_S16C_S16_Mod(mllib_s16 *dstc, const mllib_s16 *srcr, mllib_s32 order); mllib_status mllib_SignalFFT_3_S16_Mod(mllib_s16 *srcdstr, mllib_s16 *srcdsti, mllib_s32 order); mllib_status mllib_SignalFFT_3_S16C_Mod(mllib_s16 *srcdstc, mllib_s32 order); mllib_status mllib_SignalFFT_3_F32_F32(mllib_f32 *dstr, mllib_f32 *dsti, const mllib_f32 *srcr, const mllib_f32 *srci, mllib_s32 order); mllib_status mllib_SignalFFT_3_F32C_F32C(mllib_f32 *dstc, const mllib_f32 *srcc, mllib_s32 order); mllib_status mllib_SignalFFT_3_F32C_F32(mllib_f32 *dstc, const mllib_f32 *srcr, mllib_s32 order); mllib_status mllib_SignalFFT_3_F32(mllib_f32 *srcdstr, mllib_f32 *srcdsti, mllib_s32 order); mllib_status mllib_SignalFFT_3_F32C(mllib_f32 *srcdstc, mllib_s32 order); mllib_status mllib_SignalFFT_3_D64_D64(mllib_d64 *dstr, mllib_d64 *dsti, const mllib_d64 *srcr, const mllib_d64 *srci, mllib_s32 order); mllib_status mllib_SignalFFT_3_D64C_D64C(mllib_d64 *dstc, const mllib_d64 *srcc, mllib_s32 order); mllib_status mllib_SignalFFT_3_D64C_D64(mllib_d64 *dstc, const mllib_d64 *srcr, mllib_s32 order); mllib_status mllib_SignalFFT_3_D64(mllib_d64 *srcdstr, mllib_d64 *srcdsti, mllib_s32 order); mllib_status mllib_SignalFFT_3_D64C(mllib_d64 *srcdstc, mllib_s32 order);</pre>

DESCRIPTION

Each of the functions in this group performs Fast Fourier Transform (FFT).

The following equation is used for forward FFT:

$$\text{dst}[k] = \frac{1}{C1} \sum_{n=0}^{N-1} \{\text{src}[n] * \exp(-j2*\text{PI}*n*k/N)\}$$

and the following equation is used for inverse FFT (IFFT):

$$\text{dst}[n] = \frac{1}{C2} \sum_{k=0}^{N-1} \{\text{src}[k] * \exp(j2*\text{PI}*n*k/N)\}$$

where

$$\begin{aligned} k &= 0, 1, \dots, (N - 1) \\ n &= 0, 1, \dots, (N - 1) \\ N &= 2**\text{order} \end{aligned}$$

The signal FFT/IFFT functions can be categorized into four groups according to the ScaleMode in the function names in the following form:

```
mllib_Signal [FFT|IFFT]_ScaleMode_OutType_InType_OpMode ()
mllib_Signal [FFT|IFFT]_ScaleMode_DataType_OpMode ()
```

The scaling factors C1 and C2 used in the equations are defined as follows:

- For ScaleMode = 1, C1 = 1 and C2 = 2**order.
- For ScaleMode = 2, C1 = 2**order and C2 = 1.
- For ScaleMode = 3, C1 = C2 = 2** (order/2) when order is even, or C1 = 2** ((order+1)/2) and C2 = 2** ((order-1)/2) when order is odd.
- For ScaleMode = 4, C1 = 2**P and C2 = 2**Q, where P and Q are adaptive scaling factors and are generated by the functions.

For functions with only real parts for the source signal, the imaginary parts are assumed to be all zero. For functions with only real parts for the destination signal, the imaginary parts are discarded. The functions with only one data type in their names perform the operation in place.

PARAMETERS

Each function takes some of the following arguments:

<i>dst</i>	Destination signal array that contains the real parts.
<i>dsti</i>	Destination signal array that contains the imaginary parts.
<i>srcr</i>	Source signal array that contains the real parts.
<i>srci</i>	Source signal array that contains the imaginary parts.
<i>dstc</i>	Complex destination signal array. <i>dstc</i> [2*i] contains the real parts, and <i>dstc</i> [2*i+1] contains the imaginary parts.
<i>src</i>	Complex source signal array. <i>src</i> [2*i] contains the real parts, and <i>src</i> [2*i+1] contains the imaginary parts.

mllib_SignalFFT_3(3MLIB)

srcdstr Source and destination signal array that contains the real parts.
srcdsti Source and destination signal array that contains the imaginary parts.
srcdstc Complex source and destination signal array. *srcdstc*[2*i] contains the real parts, and *srcdstc*[2*i+1] contains the imaginary parts.
order Order of the transformation. The base-2 logarithm of the number of data samples.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalFFT_1(3MLIB)`, `mllib_SignalFFT_2(3MLIB)`, `mllib_SignalFFT_4(3MLIB)`, `mllib_SignalIFFT_1(3MLIB)`, `mllib_SignalIFFT_2(3MLIB)`, `mllib_SignalIFFT_3(3MLIB)`, `mllib_SignalIFFT_4(3MLIB)`, `attributes(5)`

NAME	mllib_SignalFFT_4, mllib_SignalFFT_4_S16_S16, mllib_SignalFFT_4_S16C_S16C, mllib_SignalFFT_4_S16C_S16, mllib_SignalFFT_4_S16, mllib_SignalFFT_4_S16C – signal Fast Fourier Transform (FFT)
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalFFT_4_S16_S16(mllib_s16 *dstr, mllib_s16 *dsti, const mllib_s16 *srcr, const mllib_s16 *srci, mllib_s32 order, mllib_s32 *scale); mllib_status mllib_SignalFFT_4_S16C_S16C(mllib_s16 *dstc, const mllib_s16 *srcr, mllib_s32 order, mllib_s32 *scale); mllib_status mllib_SignalFFT_4_S16C_S16(mllib_s16 *dstc, const mllib_s16 *srcr, mllib_s32 order, mllib_s32 *scale); mllib_status mllib_SignalFFT_4_S16(mllib_s16 *srcdstr, mllib_s16 *srcdsti, mllib_s32 order, mllib_s32 *scale); mllib_status mllib_SignalFFT_4_S16C(mllib_s16 *srcdstc, mllib_s32 order, mllib_s32 *scale);</pre>
DESCRIPTION	<p>Each of the functions in this group performs Fast Fourier Transform (FFT).</p> <p>The following equation is used for forward FFT:</p> $\text{dst}[k] = \frac{1}{C1} \sum_{n=0}^{N-1} \{\text{src}[n] * \exp(-j2*\text{PI}*n*k/N)\}$ <p>and the following equation is used for inverse FFT (IFFT):</p> $\text{dst}[n] = \frac{1}{C2} \sum_{k=0}^{N-1} \{\text{src}[k] * \exp(j2*\text{PI}*n*k/N)\}$ <p>where</p> <p>k = 0, 1, ..., (N - 1) n = 0, 1, ..., (N - 1) N = 2**order</p> <p>The signal FFT/IFFT functions can be categorized into four groups according to the ScaleMode in the function names in the following form:</p> <pre>mllib_Signal[FFT IFFT]_ScaleMode_OutType_InType_OpMode() mllib_Signal[FFT IFFT]_ScaleMode_DataType_OpMode()</pre> <p>The scaling factors C1 and C2 used in the equations are defined as follows:</p> <ul style="list-style-type: none"> ■ For ScaleMode = 1, C1 = 1 and C2 = 2**order. ■ For ScaleMode = 2, C1 = 2**order and C2 = 1. ■ For ScaleMode = 3, C1 = C2 = 2** (order/2) when order is even, or C1 = 2** ((order+1)/2) and C2 = 2** ((order-1)/2) when order is odd.

mllib_SignalFFT_4(3MLIB)

- For ScaleMode = 4, $C1 = 2^{**P}$ and $C2 = 2^{**Q}$, where P and Q are adaptive scaling factors and are generated by the functions.

For functions with only real parts for the source signal, the imaginary parts are assumed to be all zero. For functions with only real parts for the destination signal, the imaginary parts are discarded. The functions with only one data type in their names perform the operation in place.

PARAMETERS

Each function takes some of the following arguments:

<i>dstr</i>	Destination signal array that contains the real parts.
<i>dsti</i>	Destination signal array that contains the imaginary parts.
<i>srcr</i>	Source signal array that contains the real parts.
<i>srci</i>	Source signal array that contains the imaginary parts.
<i>dstc</i>	Complex destination signal array. <i>dstc</i> [2*i] contains the real parts, and <i>dstc</i> [2*i+1] contains the imaginary parts.
<i>src</i>	Complex source signal array. <i>src</i> [2*i] contains the real parts, and <i>src</i> [2*i+1] contains the imaginary parts.
<i>srcdstr</i>	Source and destination signal array that contains the real parts.
<i>srcdsti</i>	Source and destination signal array that contains the imaginary parts.
<i>srcdstc</i>	Complex source and destination signal array. <i>srcdstc</i> [2*i] contains the real parts, and <i>srcdstc</i> [2*i+1] contains the imaginary parts.
<i>order</i>	Order of the transformation. The base-2 logarithm of the number of data samples.
<i>scale</i>	Adaptive scaling factor.

RETURN VALUES

The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

`mllib_SignalFFT_1(3MLIB)`, `mllib_SignalFFT_2(3MLIB)`,
`mllib_SignalFFT_3(3MLIB)`, `mllib_SignalIFFT_1(3MLIB)`,
`mllib_SignalIFFT_2(3MLIB)`, `mllib_SignalIFFT_3(3MLIB)`,
`mllib_SignalIFFT_4(3MLIB)`, `attributes(5)`

NAME	mllib_SignalFFTW_1, mllib_SignalFFTW_1_S16_S16_Mod, mllib_SignalFFTW_1_S16C_S16C_Mod, mllib_SignalFFTW_1_S16C_S16_Mod, mllib_SignalFFTW_1_S16_Mod, mllib_SignalFFTW_1_S16C_Mod, mllib_SignalFFTW_1_F32_F32, mllib_SignalFFTW_1_F32C_F32C, mllib_SignalFFTW_1_F32C_F32, mllib_SignalFFTW_1_F32, mllib_SignalFFTW_1_F32C – signal Fast Fourier Transform with windowing (FFTW)
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalFFTW_1_S16_S16_Mod(mllib_s16 *dst, mllib_s16 *dsti, const mllib_s16 *srcr, const mllib_s16 *srci, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_1_S16C_S16C_Mod(mllib_s16 *dstc, const mllib_s16 *srcc, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_1_S16C_S16_Mod(mllib_s16 *dstc, const mllib_s16 *srcr, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_1_S16_Mod(mllib_s16 *srcdstr, mllib_s16 *srcdsti, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_1_S16C_Mod(mllib_s16 *srcdstc, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_1_F32_F32(mllib_f32 *dst, mllib_f32 *dsti, const mllib_f32 *srcr, const mllib_f32 *srci, const mllib_f32 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_1_F32C_F32C(mllib_f32 *dstc, const mllib_f32 *srcc, const mllib_f32 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_1_F32C_F32(mllib_f32 *dstc, const mllib_f32 *srcr, const mllib_f32 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_1_F32(mllib_f32 *srcdstr, mllib_f32 *srcdsti, const mllib_f32 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_1_F32C(mllib_f32 *srcdstc, const mllib_f32 *window, mllib_s32 order);</pre>
DESCRIPTION	<p>Each of the functions in this group performs Fast Fourier Transform with windowing (FFTW).</p> <p>The FFTW functions use the following equation:</p> $dst[k] = \frac{1}{C1} \sum_{n=0}^{N-1} \{src[n] * window[n] * exp(-j2*PI*n*k/N)\}$ <p>and the IFFTW functions use the following equation:</p> $dst[n] = \frac{1}{C2} \sum_{k=0}^{N-1} \{src[k] * window[k] * exp(j2*PI*n*k/N)\}$

mllib_SignalFFTW_1(3MLIB)

where

```
k = 0, 1, ..., (N - 1)
n = 0, 1, ..., (N - 1)
N = 2**order
```

The signal FFTW/IFFTW functions can be categorized into four groups according to the `ScaleMode` in the function names in the following form:

```
mllib_Signal[FFTW|IFFTW]_ScaleMode_OutType_InType_OpMode()
mllib_Signal[FFTW|IFFTW]_ScaleMode_DataType_OpMode()
```

The scaling factors `C1` and `C2` used in the equations are defined as follows:

- For `ScaleMode = 1`, `C1 = 1` and `C2 = 2**order`.
- For `ScaleMode = 2`, `C1 = 2**order` and `C2 = 1`.
- For `ScaleMode = 3`, `C1 = C2 = 2** (order/2)` when `order` is even, or `C1 = 2** ((order+1)/2)` and `C2 = 2** ((order-1)/2)` when `order` is odd.
- For `ScaleMode = 4`, `C1 = 2**P` and `C2 = 2**Q`, where `P` and `Q` are adaptive scaling factors and are generated by the functions.

For functions with only real parts for the source signal, the imaginary parts are assumed to be all zero. For functions with only real parts for the destination signal, the imaginary parts are discarded. The functions with only one data type in their names perform the operation in place.

PARAMETERS

Each function takes some of the following arguments:

<i>dstr</i>	Destination signal array that contains the real parts.
<i>dsti</i>	Destination signal array that contains the imaginary parts.
<i>srcr</i>	Source signal array that contains the real parts.
<i>srci</i>	Source signal array that contains the imaginary parts.
<i>dstc</i>	Complex destination signal array. <code>dstc[2*i]</code> contains the real parts, and <code>dstc[2*i+1]</code> contains the imaginary parts.
<i>srcc</i>	Complex source signal array. <code>srcc[2*i]</code> contains the real parts, and <code>srcc[2*i+1]</code> contains the imaginary parts.
<i>srcdstr</i>	Source and destination signal array that contains the real parts.
<i>srcdsti</i>	Source and destination signal array that contains the imaginary parts.
<i>srcdstc</i>	Complex source and destination signal array. <code>srcdstc[2*i]</code> contains the real parts, and <code>srcdstc[2*i+1]</code> contains the imaginary parts.
<i>window</i>	Window coefficient array with <code>2**order</code> real elements. The window coefficients are in <code>Q15</code> format for the <code>S16</code> data type, or in <code>float</code> format for the <code>F32</code> data type.

`mllib_SignalFFTW_1(3MLIB)`

order Order of the transformation. The base-2 logarithm of the number of data samples.

RETURN VALUES Each function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalFFTW_2(3MLIB)`, `mllib_SignalFFTW_3(3MLIB)`, `mllib_SignalFFTW_4(3MLIB)`, `mllib_SignalIFFTW_1(3MLIB)`, `mllib_SignalIFFTW_2(3MLIB)`, `mllib_SignalIFFTW_3(3MLIB)`, `mllib_SignalIFFTW_4(3MLIB)`, `attributes(5)`

mllib_SignalFFTW_2(3MLIB)

NAME	mllib_SignalFFTW_2, mllib_SignalFFTW_2_S16_S16, mllib_SignalFFTW_2_S16C_S16C, mllib_SignalFFTW_2_S16C_S16, mllib_SignalFFTW_2_S16, mllib_SignalFFTW_2_S16C, mllib_SignalFFTW_2_F32_F32, mllib_SignalFFTW_2_F32C_F32C, mllib_SignalFFTW_2_F32C_F32, mllib_SignalFFTW_2_F32, mllib_SignalFFTW_2_F32C – signal Fast Fourier Transform with windowing (FFTW)
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalFFTW_2_S16_S16(mllib_s16 *dstr, mllib_s16 *dsti, const mllib_s16 *srcr, const mllib_s16 *srci, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_2_S16C_S16C(mllib_s16 *dstc, const mllib_s16 *srcr, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_2_S16C_S16(mllib_s16 *dstc, const mllib_s16 *srcr, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_2_S16(mllib_s16 *srcdstr, mllib_s16 *srcdsti, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_2_S16C(mllib_s16 *srcdstc, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_2_F32_F32(mllib_f32 *dstr, mllib_f32 *dsti, const mllib_f32 *srcr, const mllib_f32 *srci, const mllib_f32 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_2_F32C_F32C(mllib_f32 *dstc, const mllib_f32 *srcr, const mllib_f32 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_2_F32C_F32(mllib_f32 *dstc, const mllib_f32 *srcr, const mllib_f32 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_2_F32(mllib_f32 *srcdstr, mllib_f32 *srcdsti, const mllib_f32 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_2_F32C(mllib_f32 *srcdstc, const mllib_f32 *window, mllib_s32 order);</pre>
DESCRIPTION	<p>Each of the functions in this group performs Fast Fourier Transform with windowing (FFTW).</p> <p>The FFTW functions use the following equation:</p> $\text{dst}[k] = \frac{1}{C1} \sum_{n=0}^{N-1} \{ \text{src}[n] * \text{window}[n] * \exp(-j2*PI*n*k/N) \}$ <p>and the IFFTW functions use the following equation:</p> $\text{dst}[n] = \frac{1}{C2} \sum_{k=0}^{N-1} \{ \text{src}[k] * \text{window}[k] * \exp(j2*PI*n*k/N) \}$ <p>where</p>


```

k = 0, 1, ..., (N - 1)
n = 0, 1, ..., (N - 1)
N = 2**order

```

The signal FFTW/IFFTW functions can be categorized into four groups according to the `ScaleMode` in the function names in the following form:

```

mllib_Signal [FFTW|IFFTW]_ScaleMode_OutType_InType_OpMode()
mllib_Signal [FFTW|IFFTW]_ScaleMode_DataType_OpMode()

```

The scaling factors `C1` and `C2` used in the equations are defined as follows:

- For `ScaleMode = 1`, `C1 = 1` and `C2 = 2**order`.
- For `ScaleMode = 2`, `C1 = 2**order` and `C2 = 1`.
- For `ScaleMode = 3`, `C1 = C2 = 2** (order/2)` when `order` is even, or `C1 = 2** ((order+1)/2)` and `C2 = 2** ((order-1)/2)` when `order` is odd.
- For `ScaleMode = 4`, `C1 = 2**P` and `C2 = 2**Q`, where `P` and `Q` are adaptive scaling factors and are generated by the functions.

For functions with only real parts for the source signal, the imaginary parts are assumed to be all zero. For functions with only real parts for the destination signal, the imaginary parts are discarded. The functions with only one data type in their names perform the operation in place.

PARAMETERS

Each function takes some of the following arguments:

<i>dst</i>	Destination signal array that contains the real parts.
<i>dsti</i>	Destination signal array that contains the imaginary parts.
<i>src</i>	Source signal array that contains the real parts.
<i>srci</i>	Source signal array that contains the imaginary parts.
<i>dstc</i>	Complex destination signal array. <code>dstc [2*i]</code> contains the real parts, and <code>dstc [2*i+1]</code> contains the imaginary parts.
<i>src</i>	Complex source signal array. <code>src [2*i]</code> contains the real parts, and <code>src [2*i+1]</code> contains the imaginary parts.
<i>srcdst</i>	Source and destination signal array that contains the real parts.
<i>srcdsti</i>	Source and destination signal array that contains the imaginary parts.
<i>srcdstc</i>	Complex source and destination signal array. <code>srcdstc [2*i]</code> contains the real parts, and <code>srcdstc [2*i+1]</code> contains the imaginary parts.
<i>window</i>	Window coefficient array with <code>2**order</code> real elements. The window coefficients are in <code>Q15</code> format for the <code>S16</code> data type, or in <code>float</code> format for the <code>F32</code> data type.
<i>order</i>	Order of the transformation. The base-2 logarithm of the number of data samples.

mllib_SignalFFTW_2(3MLIB)

RETURN VALUES Each function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalFFTW_1(3MLIB)`, `mllib_SignalFFTW_3(3MLIB)`, `mllib_SignalFFTW_4(3MLIB)`, `mllib_SignalIFFTW_1(3MLIB)`, `mllib_SignalIFFTW_2(3MLIB)`, `mllib_SignalIFFTW_3(3MLIB)`, `mllib_SignalIFFTW_4(3MLIB)`, `attributes(5)`

NAME	mllib_SignalFFTW_3, mllib_SignalFFTW_3_S16_S16_Mod, mllib_SignalFFTW_3_S16C_S16C_Mod, mllib_SignalFFTW_3_S16C_S16_Mod, mllib_SignalFFTW_3_S16_Mod, mllib_SignalFFTW_3_S16C_Mod, mllib_SignalFFTW_3_F32_F32, mllib_SignalFFTW_3_F32C_F32C, mllib_SignalFFTW_3_F32C_F32, mllib_SignalFFTW_3_F32, mllib_SignalFFTW_3_F32C – signal Fast Fourier Transform with windowing (FFTW)
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalFFTW_3_S16_S16_Mod(mllib_s16 *dst, mllib_s16 *dsti, const mllib_s16 *srcr, const mllib_s16 *srci, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_3_S16C_S16C_Mod(mllib_s16 *dstc, const mllib_s16 *srcc, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_3_S16C_S16_Mod(mllib_s16 *dstc, const mllib_s16 *srcr, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_3_S16_Mod(mllib_s16 *srcdstr, mllib_s16 *srcdsti, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_3_S16C_Mod(mllib_s16 *srcdstc, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_3_F32_F32(mllib_f32 *dst, mllib_f32 *dsti, const mllib_f32 *srcr, const mllib_f32 *srci, const mllib_f32 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_3_F32C_F32C(mllib_f32 *dstc, const mllib_f32 *srcc, const mllib_f32 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_3_F32C_F32(mllib_f32 *dstc, const mllib_f32 *srcr, const mllib_f32 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_3_F32(mllib_f32 *srcdstr, mllib_f32 *srcdsti, const mllib_f32 *window, mllib_s32 order); mllib_status mllib_SignalFFTW_3_F32C(mllib_f32 *srcdstc, const mllib_f32 *window, mllib_s32 order);</pre>
DESCRIPTION	<p>Each of the functions in this group performs Fast Fourier Transform with windowing (FFTW).</p> <p>The FFTW functions use the following equation:</p> $dst[k] = \frac{1}{C1} \sum_{n=0}^{N-1} \{src[n] * window[n] * \exp(-j2*PI*n*k/N)\}$ <p>and the IFFTW functions use the following equation:</p> $dst[n] = \frac{1}{C2} \sum_{k=0}^{N-1} \{src[k] * window[k] * \exp(j2*PI*n*k/N)\}$

mllib_SignalFFTW_3(3MLIB)

where

```
k = 0, 1, ..., (N - 1)
n = 0, 1, ..., (N - 1)
N = 2**order
```

The signal FFTW/IFFTW functions can be categorized into four groups according to the `ScaleMode` in the function names in the following form:

```
mllib_Signal[FFTW|IFFTW]_ScaleMode_OutType_InType_OpMode()
mllib_Signal[FFTW|IFFTW]_ScaleMode_DataType_OpMode()
```

The scaling factors `C1` and `C2` used in the equations are defined as follows:

- For `ScaleMode = 1`, `C1 = 1` and `C2 = 2**order`.
- For `ScaleMode = 2`, `C1 = 2**order` and `C2 = 1`.
- For `ScaleMode = 3`, `C1 = C2 = 2** (order/2)` when `order` is even, or `C1 = 2** ((order+1)/2)` and `C2 = 2** ((order-1)/2)` when `order` is odd.
- For `ScaleMode = 4`, `C1 = 2**P` and `C2 = 2**Q`, where `P` and `Q` are adaptive scaling factors and are generated by the functions.

For functions with only real parts for the source signal, the imaginary parts are assumed to be all zero. For functions with only real parts for the destination signal, the imaginary parts are discarded. The functions with only one data type in their names perform the operation in place.

PARAMETERS

Each function takes some of the following arguments:

<i>dstr</i>	Destination signal array that contains the real parts.
<i>dsti</i>	Destination signal array that contains the imaginary parts.
<i>srcr</i>	Source signal array that contains the real parts.
<i>srci</i>	Source signal array that contains the imaginary parts.
<i>dstc</i>	Complex destination signal array. <code>dstc[2*i]</code> contains the real parts, and <code>dstc[2*i+1]</code> contains the imaginary parts.
<i>srcc</i>	Complex source signal array. <code>srcc[2*i]</code> contains the real parts, and <code>srcc[2*i+1]</code> contains the imaginary parts.
<i>srcdstr</i>	Source and destination signal array that contains the real parts.
<i>srcdsti</i>	Source and destination signal array that contains the imaginary parts.
<i>srcdstc</i>	Complex source and destination signal array. <code>srcdstc[2*i]</code> contains the real parts, and <code>srcdstc[2*i+1]</code> contains the imaginary parts.
<i>window</i>	Window coefficient array with <code>2**order</code> real elements. The window coefficients are in <code>Q15</code> format for the <code>S16</code> data type, or in <code>float</code> format for the <code>F32</code> data type.

`mllib_SignalFFTW_3(3MLIB)`

order Order of the transformation. The base-2 logarithm of the number of data samples.

RETURN VALUES Each function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalFFTW_1(3MLIB)`, `mllib_SignalFFTW_2(3MLIB)`, `mllib_SignalFFTW_4(3MLIB)`, `mllib_SignalIFFTW_1(3MLIB)`, `mllib_SignalIFFTW_2(3MLIB)`, `mllib_SignalIFFTW_3(3MLIB)`, `mllib_SignalIFFTW_4(3MLIB)`, `attributes(5)`

mllib_SignalFFTW_4(3MLIB)

NAME	mllib_SignalFFTW_4, mllib_SignalFFTW_4_S16_S16, mllib_SignalFFTW_4_S16C_S16C, mllib_SignalFFTW_4_S16C_S16, mllib_SignalFFTW_4_S16, mllib_SignalFFTW_4_S16C – signal Fast Fourier Transform with windowing (FFTW)
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalFFTW_4_S16_S16(mllib_s16 *dstr, mllib_s16 *dsti, const mllib_s16 *srcr, const mllib_s16 *srci, const mllib_s16 *window, mllib_s32 order, mllib_s32 *scale); mllib_status mllib_SignalFFTW_4_S16C_S16C(mllib_s16 *dstc, const mllib_s16 *srcr, const mllib_s16 *window, mllib_s32 order, mllib_s32 *scale); mllib_status mllib_SignalFFTW_4_S16C_S16(mllib_s16 *dstc, const mllib_s16 *srcr, const mllib_s16 *window, mllib_s32 order, mllib_s32 *scale); mllib_status mllib_SignalFFTW_4_S16(mllib_s16 *srcdstr, mllib_s16 *srcdsti, const mllib_s16 *window, mllib_s32 order, mllib_s32 *scale); mllib_status mllib_SignalFFTW_4_S16C(mllib_s16 *srcdstc, const mllib_s16 *window, mllib_s32 order, mllib_s32 *scale);</pre>
DESCRIPTION	<p>Each of the functions in this group performs Fast Fourier Transform with windowing (FFTW).</p> <p>The FFTW functions use the following equation:</p> $\text{dst}[k] = \frac{1}{C1} \sum_{n=0}^{N-1} \{\text{src}[n] * \text{window}[n] * \exp(-j2*PI*n*k/N)\}$ <p>and the IFFTW functions use the following equation:</p> $\text{dst}[n] = \frac{1}{C2} \sum_{k=0}^{N-1} \{\text{src}[k] * \text{window}[k] * \exp(j2*PI*n*k/N)\}$ <p>where</p> <p>k = 0, 1, ..., (N - 1) n = 0, 1, ..., (N - 1) N = 2**order</p> <p>The signal FFTW/IFFTW functions can be categorized into four groups according to the ScaleMode in the function names in the following form:</p> <pre>mllib_Signal[FFTW IFFTW]_ScaleMode_OutType_InType_OpMode() mllib_Signal[FFTW IFFTW]_ScaleMode_DataType_OpMode()</pre> <p>The scaling factors C1 and C2 used in the equations are defined as follows:</p> <ul style="list-style-type: none">■ For ScaleMode = 1, C1 = 1 and C2 = 2**order.■ For ScaleMode = 2, C1 = 2**order and C2 = 1.

- For `ScaleMode = 3`, $C1 = C2 = 2^{(order/2)}$ when `order` is even, or $C1 = 2^{((order+1)/2)}$ and $C2 = 2^{((order-1)/2)}$ when `order` is odd.
- For `ScaleMode = 4`, $C1 = 2^P$ and $C2 = 2^Q$, where `P` and `Q` are adaptive scaling factors and are generated by the functions.

For functions with only real parts for the source signal, the imaginary parts are assumed to be all zero. For functions with only real parts for the destination signal, the imaginary parts are discarded. The functions with only one data type in their names perform the operation in place.

PARAMETERS Each function takes some of the following arguments:

<i>dstr</i>	Destination signal array that contains the real parts.
<i>dsti</i>	Destination signal array that contains the imaginary parts.
<i>srcr</i>	Source signal array that contains the real parts.
<i>srci</i>	Source signal array that contains the imaginary parts.
<i>dstc</i>	Complex destination signal array. <i>dstc</i> [2*i] contains the real parts, and <i>dstc</i> [2*i+1] contains the imaginary parts.
<i>src</i>	Complex source signal array. <i>src</i> [2*i] contains the real parts, and <i>src</i> [2*i+1] contains the imaginary parts.
<i>srcdstr</i>	Source and destination signal array that contains the real parts.
<i>srcdsti</i>	Source and destination signal array that contains the imaginary parts.
<i>srcdstc</i>	Complex source and destination signal array. <i>srcdstc</i> [2*i] contains the real parts, and <i>srcdstc</i> [2*i+1] contains the imaginary parts.
<i>window</i>	Window coefficient array with 2*order real elements. The window coefficients are in Q15 format for the S16 data type, or in float format for the F32 data type.
<i>order</i>	Order of the transformation. The base-2 logarithm of the number of data samples.
<i>scale</i>	Adaptive scaling factor.

RETURN VALUES Each function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

mllib_SignalFFTW_4(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	MT-Safe

SEE ALSO mllib_SignalFFTW_1(3MLIB), mllib_SignalFFTW_2(3MLIB),
mllib_SignalFFTW_3(3MLIB), mllib_SignalIFFTW_1(3MLIB),
mllib_SignalIFFTW_2(3MLIB), mllib_SignalIFFTW_3(3MLIB),
mllib_SignalIFFTW_4(3MLIB), attributes(5)

NAME mllib_SignalFIR_F32_F32 – Finite Impulse Response (FIR) filtering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalFIR_F32_F32(mllib_f32 *dst, const mllib_f32
    *src, void *filter, mllib_s32 n);
```

DESCRIPTION The mllib_SignalFIR_F32_F32() function applies the FIR filter to one signal packet and updates the filter state.

PARAMETERS The function takes the following arguments:

dst Output signal array.

src Input signal array.

filter Internal filter structure.

n Number of samples in the input signal array.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO attributes(5)

mllib_SignalFIR_F32S_F32S(3MLIB)

NAME mllib_SignalFIR_F32S_F32S – Finite Impulse Response (FIR) filtering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalFIR_F32S_F32S(mllib_f32 *dst, const
    mllib_f32 *src, void *filter, mllib_s32 n);
```

DESCRIPTION The `mllib_SignalFIR_F32S_F32S()` function applies the FIR filter to one signal packet and updates the filter state.

PARAMETERS The function takes the following arguments:

dst Output stereo signal array. `dst[2*i]` contains Channel 0, and `dst[2*i+1]` contains Channel 1.

src Input stereo signal array. `src[2*i]` contains Channel 0, and `src[2*i+1]` contains Channel 1.

filter Internal filter structure.

n Number of samples in the input signal array.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `attributes(5)`

NAME mllib_SignalFIRFree_F32_F32 – Finite Impulse Response (FIR) filtering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_SignalFIRFree_F32_F32(void *filter);
```

DESCRIPTION The mllib_SignalFIRFree_F32_F32() function releases the memory allocated for the internal filter structure.

PARAMETERS The function takes the following arguments:
filter Internal filter structure.

RETURN VALUES None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO attributes(5)

mllib_SignalFIRFree_F32S_F32S(3MLIB)

NAME | mllib_SignalFIRFree_F32S_F32S – Finite Impulse Response (FIR) filtering

SYNOPSIS | cc [*flag...*] *file...* -lmllib [*library...*]
#include <mllib.h>

void **mllib_SignalFIRFree_F32S_F32S**(void **filter*) ;

DESCRIPTION | The mllib_SignalFIRFree_F32S_F32S() function releases the memory allocated for the internal filter structure.

PARAMETERS | The function takes the following arguments:
filter Internal filter structure.

RETURN VALUES | None.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | attributes(5)

NAME mllib_SignalFIRFree_S16_S16, mllib_SignalFIRFree_S16S_S16S – Finite Impulse Response (FIR) filtering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_SignalFIRFree_S16_S16(void *filter) ;
void mllib_SignalFIRFree_S16S_S16S(void *filter) ;
```

DESCRIPTION Each of these functions releases the memory allocated for the internal filter structure.

PARAMETERS Each of the functions takes the following arguments:
filter Internal filter structure.

RETURN VALUES None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalFIR_S16_S16_Sat(3MLIB),
mllib_SignalFIRInit_S16_S16(3MLIB), attributes(5)

mllib_SignalFIRInit_F32_F32(3MLIB)

NAME | mllib_SignalFIRInit_F32_F32 – Finite Impulse Response (FIR) filtering

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalFIRInit_F32_F32(void **filter, const
mllib_f32 *flt, mllib_s32 tap);
```

DESCRIPTION | The mllib_SignalFIRInit_F32_F32() function allocates memory for the internal filter structure and converts the filter coefficients to an internal representation.

PARAMETERS | The function takes the following arguments:

filter | Internal filter structure.

flt | Filter coefficient array.

tap | Taps of the filter.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | attributes(5)

NAME mllib_SignalFIRInit_F32S_F32S – Finite Impulse Response (FIR) filtering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalFIRInit_F32S_F32S(void **filter, const
mllib_f32 *flt, mllib_s32 tap);
```

DESCRIPTION The mllib_SignalFIRInit_F32S_F32S() function allocates memory for the internal filter structure and converts the filter coefficients to an internal representation.

PARAMETERS The function takes the following arguments:

filter Internal filter structure.

flt Filter coefficient array in stereo format. flt[2*i] contains Channel 0, and flt[2*i+1] contains Channel 1

tap Taps of the filter.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO attributes(5)

mllib_SignalFIRInit_S16_S16(3MLIB)

NAME mllib_SignalFIRInit_S16_S16, mllib_SignalFIRInit_S16S_S16S – Finite Impulse Response (FIR) filtering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalFIRInit_S16_S16(void **filter, const
    mllib_f32 *flt, mllib_s32 tap);

mllib_status mllib_SignalFIRInit_S16S_S16S(void **filter, const
    mllib_f32 *flt, mllib_s32 tap);
```

DESCRIPTION Each of these functions allocates memory for the internal filter structure and converts the filter coefficients to an internal representation.

PARAMETERS Each of the functions takes the following arguments:

filter Internal filter structure.

flt Filter coefficient array.

tap Taps of the filter.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalFIR_S16_S16_Sat(3MLIB),
mllib_SignalFIRFree_S16_S16(3MLIB), attributes(5)

NAME mllib_SignalFIR_S16_S16_Sat, mllib_SignalFIR_S16S_S16S_Sat – Finite Impulse Response (FIR) filtering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalFIR_S16_S16_Sat(mllib_s16 *dst, const
    mllib_s16 *src, void *filter, mllib_s32 n);

mllib_status mllib_SignalFIR_S16S_S16S_Sat(mllib_s16 *dst, const
    mllib_s16 *src, void *filter, mllib_s32 n);
```

DESCRIPTION Each of these functions applies the FIR filter to one signal packet and updates the filter state.

PARAMETERS Each of the functions takes the following arguments:

dst Output signal array.

src Input signal array.

filter Internal filter structure.

n Number of samples in the input signal array.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_SignalFIRFree_S16_S16\(3MLIB\)](#),
[mllib_SignalFIRInit_S16_S16\(3MLIB\)](#), [attributes\(5\)](#)

mllib_SignalGaussNoise_F32(3MLIB)

NAME | mllib_SignalGaussNoise_F32 – Gaussian noise generation

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalGaussNoise_F32(mllib_f32 *gnoise, void *state,
mllib_s32 n);
```

DESCRIPTION | The `mllib_SignalGaussNoise_F32()` function generates one packet of Gaussian noise and updates the internal state.

PARAMETERS | The function takes the following arguments:

<i>gnoise</i>	Generated Gaussian noise array.
<i>state</i>	Internal state structure.
<i>n</i>	Length of the generated Gaussian wave array in number of samples.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `attributes(5)`

NAME mllib_SignalGaussNoiseFree_F32 – Gaussian noise generation

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_SignalGaussNoiseFree_F32(void *state);
```

DESCRIPTION The mllib_SignalGaussNoiseFree_F32() function releases the memory allocated for the internal state's structure.

PARAMETERS The function takes the following arguments:
state Internal state structure.

RETURN VALUES None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO attributes(5)

mllib_SignalGaussNoiseFree_S16(3MLIB)

NAME | mllib_SignalGaussNoiseFree_S16 – Gaussian noise generation

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
void mllib_SignalGaussNoiseFree_S16(void *state);
```

DESCRIPTION | The mllib_SignalGaussNoiseFree_S16() function releases the memory allocated for the internal state's structure.

PARAMETERS | The function takes the following arguments:
state Internal state structure.

RETURN VALUES | None.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_SignalGaussNoise_S16\(3MLIB\)](#),
[mllib_SignalGaussNoiseInit_S16\(3MLIB\)](#), [attributes\(5\)](#)

NAME mllib_SignalGaussNoiseInit_F32 – Gaussian noise generation

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalGaussNoiseInit_F32(void **state, mllib_f32
      mag, mllib_f32 mean, mllib_f32 stddev, mllib_f32 seed);
```

DESCRIPTION The `mllib_SignalGaussNoiseInit_F32()` function allocates memory for an internal state structure and converts the parameters into an internal representation.

PARAMETERS The function takes the following arguments:

state Internal state structure.

mag Magnitude of the Gaussian noise to be generated, in Q15 format.

mean Mean of the Gaussian noise.

stddev Standard deviation of the Gaussian noise.

seed Seed value for the pseudorandom number generator.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `attributes(5)`

mllib_SignalGaussNoiseInit_S16(3MLIB)

NAME	mllib_SignalGaussNoiseInit_S16 – Gaussian noise generation										
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalGaussNoiseInit_S16(void **<i>state</i>, mllib_s16 <i>mag</i>, mllib_f32 <i>mean</i>, mllib_f32 <i>stddev</i>, mllib_s16 <i>seed</i>);</pre>										
DESCRIPTION	The <code>mllib_SignalGaussNoiseInit_S16()</code> function allocates memory for an internal state structure and converts the parameters into an internal representation.										
PARAMETERS	The function takes the following arguments: <table><tr><td><i>state</i></td><td>Internal state structure.</td></tr><tr><td><i>mag</i></td><td>Magnitude of the Gaussian noise to be generated, in Q15 format.</td></tr><tr><td><i>mean</i></td><td>Mean of the Gaussian noise.</td></tr><tr><td><i>stddev</i></td><td>Standard deviation of the Gaussian noise.</td></tr><tr><td><i>seed</i></td><td>Seed value for the pseudorandom number generator.</td></tr></table>	<i>state</i>	Internal state structure.	<i>mag</i>	Magnitude of the Gaussian noise to be generated, in Q15 format.	<i>mean</i>	Mean of the Gaussian noise.	<i>stddev</i>	Standard deviation of the Gaussian noise.	<i>seed</i>	Seed value for the pseudorandom number generator.
<i>state</i>	Internal state structure.										
<i>mag</i>	Magnitude of the Gaussian noise to be generated, in Q15 format.										
<i>mean</i>	Mean of the Gaussian noise.										
<i>stddev</i>	Standard deviation of the Gaussian noise.										
<i>seed</i>	Seed value for the pseudorandom number generator.										
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .										
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe				
ATTRIBUTE TYPE	ATTRIBUTE VALUE										
Interface Stability	Evolving										
MT-Level	MT-Safe										
SEE ALSO	<code>mllib_SignalGaussNoise_S16(3MLIB)</code> , <code>mllib_SignalGaussNoiseFree_S16(3MLIB)</code> , <code>attributes(5)</code>										

NAME mllib_SignalGaussNoise_S16 – Gaussian noise generation

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalGaussNoise_S16(mllib_s16 *gnoise, void *state,
mllib_s32 n);
```

DESCRIPTION The mllib_SignalGaussNoise_S16() function generates one packet of Gaussian noise and updates the internal state.

PARAMETERS The function takes the following arguments:

gnoise Generated Gaussian noise array.

state Internal state structure.

n Length of the generated Gaussian wave array in number of samples.

RETURN VALUES The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_SignalGaussNoiseFree_S16\(3MLIB\)](#),
[mllib_SignalGaussNoiseInit_S16\(3MLIB\)](#), [attributes\(5\)](#)

mllib_SignalGenBartlett_F32(3MLIB)

NAME | mllib_SignalGenBartlett_F32 – Bartlett window generation

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalGenBartlett_F32(mllib_f32 *window, mllib_s32
n);
```

DESCRIPTION | The mllib_SignalGenBartlett_F32() function generates the normalized coefficients of the Bartlett window.

PARAMETERS | The function takes the following arguments:

window | Generated window coefficient array.

n | Length of window array.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | attributes(5)

NAME mllib_SignalGenBartlett_S16 – Bartlett window generation

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalGenBartlett_S16(mllib_s16 *window, mllib_s32
n);
```

DESCRIPTION The mllib_SignalGenBartlett_S16() function generates the normalized coefficients of the Bartlett window.

PARAMETERS The function takes the following arguments:

window Generated window coefficient array. The window coefficients are in Q15 format.

n Length of window array.

RETURN VALUES The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalGenHanning_S16(3MLIB), mllib_SignalGenHamming_S16(3MLIB), mllib_SignalGenBlackman_S16(3MLIB), mllib_SignalGenKaiser_S16(3MLIB), attributes(5)

mllib_SignalGenBlackman_F32(3MLIB)

NAME | mllib_SignalGenBlackman_F32 – Blackman window generation

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalGenBlackman_F32(mllib_f32 *window, mllib_f32
    alpha, mllib_s32 n);
```

DESCRIPTION | The `mllib_SignalGenBlackman_F32()` function generates the normalized coefficients of the Blackman window.

PARAMETERS | The function takes the following arguments:

<i>window</i>	Generated window coefficient array.
<i>alpha</i>	Blackman window parameter. $-1 < \alpha < 0$.
<i>n</i>	Length of window array.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `attributes(5)`

NAME mllib_SignalGenBlackman_S16 – Blackman window generation

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalGenBlackman_S16(mllib_s16 *window, mllib_f32
    alpha, mllib_s32 n);
```

DESCRIPTION The mllib_SignalGenBlackman_S16() function generates the normalized coefficients of the Blackman window.

PARAMETERS The function takes the following arguments:

window Generated window coefficient array. The window coefficients are in Q15 format.

alpha Blackman window parameter. $-1 < \alpha < 0$.

n Length of window array.

RETURN VALUES The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalGenBartlett_S16(3MLIB), mllib_SignalGenHanning_S16(3MLIB), mllib_SignalGenHamming_S16(3MLIB), mllib_SignalGenKaiser_S16(3MLIB), attributes(5)

mllib_SignalGenHamming_F32(3MLIB)

NAME	mllib_SignalGenHamming_F32 – Hamming window generation						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalGenHamming_F32(mllib_f32 *window, mllib_s32 n);</pre>						
DESCRIPTION	The <code>mllib_SignalGenHamming_F32()</code> function generates the normalized coefficients of the Hamming window.						
PARAMETERS	The function takes the following arguments: <i>window</i> Generated window coefficient array. <i>n</i> Length of window array.						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>attributes(5)</code>						

mllib_SignalGenHamming_S16(3MLIB)

NAME | mllib_SignalGenHamming_S16 – Hamming window generation

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
mllib_status mllib_SignalGenHamming_S16(mllib_s16 *window, mllib_s32  
    n);
```

DESCRIPTION | The `mllib_SignalGenHamming_S16()` function generates the normalized coefficients of the Hamming window.

PARAMETERS | The function takes the following arguments:

window | Generated window coefficient array. The window coefficients are in Q15 format.

n | Length of window array.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_SignalGenBartlett_S16(3MLIB)`,
`mllib_SignalGenHanning_S16(3MLIB)`,
`mllib_SignalGenBlackman_S16(3MLIB)`, `mllib_SignalGenKaiser_S16(3MLIB)`,
`attributes(5)`

mllib_SignalGenHanning_F32(3MLIB)

NAME | mllib_SignalGenHanning_F32 – Hanning window generation

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
mllib_status mllib_SignalGenHanning_F32(mllib_f32 *window, mllib_s32  
          n);
```

DESCRIPTION | The `mllib_SignalGenHanning_F32()` function generates the normalized coefficients of the Hanning window.

PARAMETERS | The function takes the following arguments:

window Generated window coefficient array. The window coefficients are in Q15 format.

n Length of window array.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `attributes(5)`

mllib_SignalGenHanning_S16(3MLIB)

NAME mllib_SignalGenHanning_S16 – Hanning window generation

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalGenHanning_S16(mllib_s16 *window, mllib_s32
n);
```

DESCRIPTION The mllib_SignalGenHanning_S16() function generates the normalized coefficients of the Hanning window.

PARAMETERS The function takes the following arguments:

window Generated window coefficient array. The window coefficients are in Q15 format.

n Length of window array.

RETURN VALUES The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalGenBartlett_S16(3MLIB), mllib_SignalGenHamming_S16(3MLIB), mllib_SignalGenBlackman_S16(3MLIB), mllib_SignalGenKaiser_S16(3MLIB), attributes(5)

mllib_SignalGenKaiser_F32(3MLIB)

NAME | mllib_SignalGenKaiser_F32 – Kaiser window generation

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
mllib_status mllib_SignalGenKaiser_F32(mllib_f32 *window, mllib_f32  
          beta, mllib_s32 n);
```

DESCRIPTION | The `mllib_SignalGenKaiser_F32()` function generates the normalized coefficients of the Kaiser window.

PARAMETERS | The function takes the following arguments:

<i>window</i>	Generated window coefficient array. The window coefficients are in Q15 format.
<i>beta</i>	Kaiser window parameter.
<i>n</i>	Length of window array.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `attributes(5)`

NAME mllib_SignalGenKaiser_S16 – Kaiser window generation

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalGenKaiser_S16(mllib_s16 *window, mllib_f32
    beta, mllib_s32 n);
```

DESCRIPTION The mllib_SignalGenKaiser_S16() function generates the normalized coefficients of the Kaiser window.

PARAMETERS The function takes the following arguments:

window Generated window coefficient array. The window coefficients are in Q15 format.

beta Kaiser window parameter.

n Length of window array.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_SignalGenBartlett_S16\(3MLIB\)](#), [mllib_SignalGenHanning_S16\(3MLIB\)](#), [mllib_SignalGenHamming_S16\(3MLIB\)](#), [mllib_SignalGenBlackman_S16\(3MLIB\)](#), [attributes\(5\)](#)

mllib_SignalIFFT_1(3MLIB)

NAME	mllib_SignalIFFT_1, mllib_SignalIFFT_1_S16_S16, mllib_SignalIFFT_1_S16C_S16C, mllib_SignalIFFT_1_S16_S16C, mllib_SignalIFFT_1_S16, mllib_SignalIFFT_1_S16C, mllib_SignalIFFT_1_F32_F32, mllib_SignalIFFT_1_F32C_F32C, mllib_SignalIFFT_1_F32_F32C, mllib_SignalIFFT_1_F32, mllib_SignalIFFT_1_F32C, mllib_SignalIFFT_1_D64_D64, mllib_SignalIFFT_1_D64C_D64C, mllib_SignalIFFT_1_D64_D64C, mllib_SignalIFFT_1_D64, mllib_SignalIFFT_1_D64C - signal Inverse Fast Fourier Transform (IFFT)
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmlib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalIFFT_1_S16_S16(mllib_s16 *dstr, mllib_s16 *dsti, const mllib_s16 *srcr, const mllib_s16 *srci, mllib_s32 order); mllib_status mllib_SignalIFFT_1_S16C_S16C(mllib_s16 *dstc, const mllib_s16 *srcc, mllib_s32 order); mllib_status mllib_SignalIFFT_1_S16_S16C(mllib_s16 *dstr, const mllib_s16 *srcc, mllib_s32 order); mllib_status mllib_SignalIFFT_1_S16(mllib_s16 *srcdstr, mllib_s16 *srcdsti, mllib_s32 order); mllib_status mllib_SignalIFFT_1_S16C(mllib_s16 *srcdstc, mllib_s32 order); mllib_status mllib_SignalIFFT_1_F32_F32(mllib_f32 *dstr, mllib_f32 *dsti, const mllib_f32 *srcr, const mllib_f32 *srci, mllib_s32 order); mllib_status mllib_SignalIFFT_1_F32C_F32C(mllib_f32 *dstc, const mllib_f32 *srcc, mllib_s32 order); mllib_status mllib_SignalIFFT_1_F32_F32C(mllib_f32 *dstr, const mllib_f32 *srcc, mllib_s32 order); mllib_status mllib_SignalIFFT_1_F32(mllib_f32 *srcdstr, mllib_f32 *srcdsti, mllib_s32 order); mllib_status mllib_SignalIFFT_1_F32C(mllib_f32 *srcdstc, mllib_s32 order); mllib_status mllib_SignalIFFT_1_D64_D64(mllib_d64 *dstr, mllib_d64 *dsti, const mllib_d64 *srcr, const mllib_d64 *srci, mllib_s32 order); mllib_status mllib_SignalIFFT_1_D64C_D64C(mllib_d64 *dstc, const mllib_d64 *srcc, mllib_s32 order); mllib_status mllib_SignalIFFT_1_D64_D64C(mllib_d64 *dstr, const mllib_d64 *srcc, mllib_s32 order); mllib_status mllib_SignalIFFT_1_D64(mllib_d64 *srcdstr, mllib_d64 *srcdsti, mllib_s32 order); mllib_status mllib_SignalIFFT_1_D64C(mllib_d64 *srcdstc, mllib_s32 order);</pre>
DESCRIPTION	Each of the functions in this group performs Inverse Fast Fourier Transform (IFFT).

The following equation is used for forward FFT:

$$\text{dst}[k] = \frac{1}{C1} \sum_{n=0}^{N-1} \{\text{src}[n] * \exp(-j2\pi n*k/N)\}$$

and the following equation is used for inverse FFT (IFFT):

$$\text{dst}[n] = \frac{1}{C2} \sum_{k=0}^{N-1} \{\text{src}[k] * \exp(j2\pi n*k/N)\}$$

where

$$\begin{aligned} k &= 0, 1, \dots, (N - 1) \\ n &= 0, 1, \dots, (N - 1) \\ N &= 2^{**}\text{order} \end{aligned}$$

The signal FFT/IFFT functions can be categorized into four groups according to the ScaleMode in the function names in the following form:

```
mllib_Signal[FFT|IFFT]_ScaleMode_OutType_InType_OpMode()
mllib_Signal[FFT|IFFT]_ScaleMode_DataType_OpMode()
```

The scaling factors C1 and C2 used in the equations are defined as follows:

- For ScaleMode = 1, C1 = 1 and C2 = 2**order.
- For ScaleMode = 2, C1 = 2**order and C2 = 1.
- For ScaleMode = 3, C1 = C2 = 2** (order/2) when order is even, or C1 = 2** ((order+1)/2) and C2 = 2** ((order-1)/2) when order is odd.
- For ScaleMode = 4, C1 = 2**P and C2 = 2**Q, where P and Q are adaptive scaling factors and are generated by the functions.

For functions with only real parts for the source signal, the imaginary parts are assumed to be all zero. For functions with only real parts for the destination signal, the imaginary parts are discarded. The functions with only one data type in their names perform the operation in place.

PARAMETERS

Each function takes some of the following arguments:

<i>dstr</i>	Destination signal array that contains the real parts.
<i>dsti</i>	Destination signal array that contains the imaginary parts.
<i>srcr</i>	Source signal array that contains the real parts.
<i>srci</i>	Source signal array that contains the imaginary parts.
<i>dstc</i>	Complex destination signal array. <i>dstc[2*i]</i> contains the real parts, and <i>dstc[2*i+1]</i> contains the imaginary parts.
<i>src</i>	Complex source signal array. <i>src[2*i]</i> contains the real parts, and <i>src[2*i+1]</i> contains the imaginary parts.
<i>srcdstr</i>	Source and destination signal array that contains the real parts.

`mllib_SignalIFFT_1(3MLIB)`

srcdsti Source and destination signal array that contains the imaginary parts.

srcdstc Complex source and destination signal array. *srcdstc*[2*i] contains the real parts, and *srcdstc*[2*i+1] contains the imaginary parts.

order Order of the transformation. The base-2 logarithm of the number of data samples.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalFFT_1(3MLIB)`, `mllib_SignalFFT_2(3MLIB)`, `mllib_SignalFFT_3(3MLIB)`, `mllib_SignalFFT_4(3MLIB)`, `mllib_SignalIFFT_2(3MLIB)`, `mllib_SignalIFFT_3(3MLIB)`, `mllib_SignalIFFT_4(3MLIB)`, `attributes(5)`

NAME	mllib_SignalIFFT_2, mllib_SignalIFFT_2_S16_S16_Mod, mllib_SignalIFFT_2_S16C_S16C_Mod, mllib_SignalIFFT_2_S16_S16C_Mod, mllib_SignalIFFT_2_S16_Mod, mllib_SignalIFFT_2_S16C_Mod, mllib_SignalIFFT_2_F32_F32, mllib_SignalIFFT_2_F32C_F32C, mllib_SignalIFFT_2_F32_F32C, mllib_SignalIFFT_2_F32, mllib_SignalIFFT_2_F32C, mllib_SignalIFFT_2_D64_D64, mllib_SignalIFFT_2_D64C_D64C, mllib_SignalIFFT_2_D64_D64C, mllib_SignalIFFT_2_D64, mllib_SignalIFFT_2_D64C – signal Inverse Fast Fourier Transform (IFFT)
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalIFFT_2_S16_S16_Mod(mllib_s16 *dstr, mllib_s16 *dsti, const mllib_s16 *srcr, const mllib_s16 *srci, mllib_s32 order); mllib_status mllib_SignalIFFT_2_S16C_S16C_Mod(mllib_s16 *dstc, const mllib_s16 *srcc, mllib_s32 order); mllib_status mllib_SignalIFFT_2_S16_S16C_Mod(mllib_s16 *dstr, const mllib_s16 *srcc, mllib_s32 order); mllib_status mllib_SignalIFFT_2_S16_Mod(mllib_s16 *srcdstr, mllib_s16 *srcdsti, mllib_s32 order); mllib_status mllib_SignalIFFT_2_S16C_Mod(mllib_s16 *srcdstc, mllib_s32 order); mllib_status mllib_SignalIFFT_2_F32_F32(mllib_f32 *dstr, mllib_f32 *dsti, const mllib_f32 *srcr, const mllib_f32 *srci, mllib_s32 order); mllib_status mllib_SignalIFFT_2_F32C_F32C(mllib_f32 *dstc, const mllib_f32 *srcc, mllib_s32 order); mllib_status mllib_SignalIFFT_2_F32_F32C(mllib_f32 *dstr, const mllib_f32 *srcc, mllib_s32 order); mllib_status mllib_SignalIFFT_2_F32(mllib_f32 *srcdstr, mllib_f32 *srcdsti, mllib_s32 order); mllib_status mllib_SignalIFFT_2_F32C(mllib_f32 *srcdstc, mllib_s32 order); mllib_status mllib_SignalIFFT_2_D64_D64(mllib_d64 *dstr, mllib_d64 *dsti, const mllib_d64 *srcr, const mllib_d64 *srci, mllib_s32 order); mllib_status mllib_SignalIFFT_2_D64C_D64C(mllib_d64 *dstc, const mllib_d64 *srcc, mllib_s32 order); mllib_status mllib_SignalIFFT_2_D64_D64C(mllib_d64 *dstr, const mllib_d64 *srcc, mllib_s32 order); mllib_status mllib_SignalIFFT_2_D64(mllib_d64 *srcdstr, mllib_d64 *srcdsti, mllib_s32 order); mllib_status mllib_SignalIFFT_2_D64C(mllib_d64 *srcdstc, mllib_s32 order);</pre>

mllib_SignalIFFT_2(3MLIB)

DESCRIPTION Each of the functions in this group performs Inverse Fast Fourier Transform (IFFT).

The following equation is used for forward FFT:

$$\text{dst}[k] = \frac{1}{C1} \sum_{n=0}^{N-1} \{\text{src}[n] * \exp(-j2*\text{PI}*n*k/N)\}$$

and the following equation is used for inverse FFT (IFFT):

$$\text{dst}[n] = \frac{1}{C2} \sum_{k=0}^{N-1} \{\text{src}[k] * \exp(j2*\text{PI}*n*k/N)\}$$

where

$$\begin{aligned} k &= 0, 1, \dots, (N - 1) \\ n &= 0, 1, \dots, (N - 1) \\ N &= 2**\text{order} \end{aligned}$$

The signal FFT/IFFT functions can be categorized into four groups according to the ScaleMode in the function names in the following form:

```
mllib_Signal [FFT|IFFT]_ScaleMode_OutType_InType_OpMode ()  
mllib_Signal [FFT|IFFT]_ScaleMode_DataType_OpMode ()
```

The scaling factors C1 and C2 used in the equations are defined as follows:

- For ScaleMode = 1, C1 = 1 and C2 = 2**order.
- For ScaleMode = 2, C1 = 2**order and C2 = 1.
- For ScaleMode = 3, C1 = C2 = 2** (order/2) when order is even, or C1 = 2** ((order+1)/2) and C2 = 2** ((order-1)/2) when order is odd.
- For ScaleMode = 4, C1 = 2**P and C2 = 2**Q, where P and Q are adaptive scaling factors and are generated by the functions.

For functions with only real parts for the source signal, the imaginary parts are assumed to be all zero. For functions with only real parts for the destination signal, the imaginary parts are discarded. The functions with only one data type in their names perform the operation in place.

PARAMETERS Each function takes some of the following arguments:

<i>dstr</i>	Destination signal array that contains the real parts.
<i>dsti</i>	Destination signal array that contains the imaginary parts.
<i>srcr</i>	Source signal array that contains the real parts.
<i>srci</i>	Source signal array that contains the imaginary parts.
<i>dstc</i>	Complex destination signal array. <i>dstc</i> [2*i] contains the real parts, and <i>dstc</i> [2*i+1] contains the imaginary parts.
<i>src</i>	Complex source signal array. <i>src</i> [2*i] contains the real parts, and <i>src</i> [2*i+1] contains the imaginary parts.

mllib_SignalIFFT_2(3MLIB)

srcdstr Source and destination signal array that contains the real parts.
srcdsti Source and destination signal array that contains the imaginary parts.
srcdstc Complex source and destination signal array. *srcdstc*[2*i] contains the real parts, and *srcdstc*[2*i+1] contains the imaginary parts.
order Order of the transformation. The base-2 logarithm of the number of data samples.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalFFT_1(3MLIB)`, `mllib_SignalFFT_2(3MLIB)`,
`mllib_SignalFFT_3(3MLIB)`, `mllib_SignalFFT_4(3MLIB)`,
`mllib_SignalIFFT_1(3MLIB)`, `mllib_SignalIFFT_3(3MLIB)`,
`mllib_SignalIFFT_4(3MLIB)`, `attributes(5)`

mllib_SignalIFFT_3(3MLIB)

NAME	mllib_SignalIFFT_3, mllib_SignalIFFT_3_S16_S16_Mod, mllib_SignalIFFT_3_S16C_S16C_Mod, mllib_SignalIFFT_3_S16_S16C_Mod, mllib_SignalIFFT_3_S16_Mod, mllib_SignalIFFT_3_S16C_Mod, mllib_SignalIFFT_3_F32_F32, mllib_SignalIFFT_3_F32C_F32C, mllib_SignalIFFT_3_F32_F32C, mllib_SignalIFFT_3_F32, mllib_SignalIFFT_3_F32C, mllib_SignalIFFT_3_D64_D64, mllib_SignalIFFT_3_D64C_D64C, mllib_SignalIFFT_3_D64_D64C, mllib_SignalIFFT_3_D64, mllib_SignalIFFT_3_D64C – signal Inverse Fast Fourier Transform (IFFT)
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalIFFT_3_S16_S16_Mod(mllib_s16 *dstr, mllib_s16 *dsti, const mllib_s16 *srcr, const mllib_s16 *srci, mllib_s32 order); mllib_status mllib_SignalIFFT_3_S16C_S16C_Mod(mllib_s16 *dstc, const mllib_s16 *srcc, mllib_s32 order); mllib_status mllib_SignalIFFT_3_S16_S16C_Mod(mllib_s16 *dstr, const mllib_s16 *srcc, mllib_s32 order); mllib_status mllib_SignalIFFT_3_S16_Mod(mllib_s16 *srcdstr, mllib_s16 *srcdsti, mllib_s32 order); mllib_status mllib_SignalIFFT_3_S16C_Mod(mllib_s16 *srcdstc, mllib_s32 order); mllib_status mllib_SignalIFFT_3_F32_F32(mllib_f32 *dstr, mllib_f32 *dsti, const mllib_f32 *srcr, const mllib_f32 *srci, mllib_s32 order); mllib_status mllib_SignalIFFT_3_F32C_F32C(mllib_f32 *dstc, const mllib_f32 *srcc, mllib_s32 order); mllib_status mllib_SignalIFFT_3_F32_F32C(mllib_f32 *dstr, const mllib_f32 *srcc, mllib_s32 order); mllib_status mllib_SignalIFFT_3_F32(mllib_f32 *srcdstr, mllib_f32 *srcdsti, mllib_s32 order); mllib_status mllib_SignalIFFT_3_F32C(mllib_f32 *srcdstc, mllib_s32 order); mllib_status mllib_SignalIFFT_3_D64_D64(mllib_d64 *dstr, mllib_d64 *dsti, const mllib_d64 *srcr, const mllib_d64 *srci, mllib_s32 order); mllib_status mllib_SignalIFFT_3_D64C_D64C(mllib_d64 *dstc, const mllib_d64 *srcc, mllib_s32 order); mllib_status mllib_SignalIFFT_3_D64_D64C(mllib_d64 *dstr, const mllib_d64 *srcc, mllib_s32 order); mllib_status mllib_SignalIFFT_3_D64(mllib_d64 *srcdstr, mllib_d64 *srcdsti, mllib_s32 order); mllib_status mllib_SignalIFFT_3_D64C(mllib_d64 *srcdstc, mllib_s32 order);</pre>

DESCRIPTION	<p>Each of the functions in this group performs Inverse Fast Fourier Transform (IFFT).</p> <p>The following equation is used for forward FFT:</p> $\text{dst}[k] = \frac{1}{C1} \sum_{n=0}^{N-1} \{\text{src}[n] * \exp(-j2*PI*n*k/N)\}$ <p>and the following equation is used for inverse FFT (IFFT):</p> $\text{dst}[n] = \frac{1}{C2} \sum_{k=0}^{N-1} \{\text{src}[k] * \exp(j2*PI*n*k/N)\}$ <p>where</p> $k = 0, 1, \dots, (N - 1)$ $n = 0, 1, \dots, (N - 1)$ $N = 2**\text{order}$ <p>The signal FFT/IFFT functions can be categorized into four groups according to the ScaleMode in the function names in the following form:</p> <pre>mllib_Signal [FFT IFFT]_ScaleMode_OutType_InType_OpMode () mllib_Signal [FFT IFFT]_ScaleMode_DataType_OpMode ()</pre> <p>The scaling factors C1 and C2 used in the equations are defined as follows:</p> <ul style="list-style-type: none"> ■ For ScaleMode = 1, C1 = 1 and C2 = 2**order. ■ For ScaleMode = 2, C1 = 2**order and C2 = 1. ■ For ScaleMode = 3, C1 = C2 = 2** (order/2) when order is even, or C1 = 2** ((order+1)/2) and C2 = 2** ((order-1)/2) when order is odd. ■ For ScaleMode = 4, C1 = 2**P and C2 = 2**Q, where P and Q are adaptive scaling factors and are generated by the functions. <p>For functions with only real parts for the source signal, the imaginary parts are assumed to be all zero. For functions with only real parts for the destination signal, the imaginary parts are discarded. The functions with only one data type in their names perform the operation in place.</p>
PARAMETERS	<p>Each function takes some of the following arguments:</p> <p><i>dst</i> Destination signal array that contains the real parts.</p> <p><i>dsti</i> Destination signal array that contains the imaginary parts.</p> <p><i>srcr</i> Source signal array that contains the real parts.</p> <p><i>srci</i> Source signal array that contains the imaginary parts.</p> <p><i>dstc</i> Complex destination signal array. <i>dstc</i>[2*i] contains the real parts, and <i>dstc</i>[2*i+1] contains the imaginary parts.</p> <p><i>src</i> Complex source signal array. <i>src</i>[2*i] contains the real parts, and <i>src</i>[2*i+1] contains the imaginary parts.</p>

mllib_SignalIFFT_3(3MLIB)

<i>srcdstr</i>	Source and destination signal array that contains the real parts.
<i>srcdsti</i>	Source and destination signal array that contains the imaginary parts.
<i>srcdstc</i>	Complex source and destination signal array. <i>srcdstc</i> [2*i] contains the real parts, and <i>srcdstc</i> [2*i+1] contains the imaginary parts.
<i>order</i>	Order of the transformation. The base-2 logarithm of the number of data samples.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalFFT_1(3MLIB)`, `mllib_SignalFFT_2(3MLIB)`,
`mllib_SignalFFT_3(3MLIB)`, `mllib_SignalFFT_4(3MLIB)`,
`mllib_SignalIFFT_1(3MLIB)`, `mllib_SignalIFFT_2(3MLIB)`,
`mllib_SignalIFFT_4(3MLIB)`, `attributes(5)`

NAME	mllib_SignalIFFT_4, mllib_SignalIFFT_4_S16_S16, mllib_SignalIFFT_4_S16C_S16C, mllib_SignalIFFT_4_S16_S16C, mllib_SignalIFFT_4_S16, mllib_SignalIFFT_4_S16C – signal Inverse Fast Fourier Transform (IFFT)
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalIFFT_4_S16_S16(mllib_s16 *dstr, mllib_s16 *dsti, const mllib_s16 *srcr, const mllib_s16 *srci, mllib_s32 order, mllib_s32 *scale); mllib_status mllib_SignalIFFT_4_S16C_S16C(mllib_s16 *dstc, const mllib_s16 *srcc, mllib_s32 order, mllib_s32 *scale); mllib_status mllib_SignalIFFT_4_S16_S16C(mllib_s16 *dstr, const mllib_s16 *srcc, mllib_s32 order, mllib_s32 *scale); mllib_status mllib_SignalIFFT_4_S16(mllib_s16 *srcdstr, mllib_s16 *srcdsti, mllib_s32 order, mllib_s32 *scale); mllib_status mllib_SignalIFFT_4_S16C(mllib_s16 *srcdstc, mllib_s32 order, mllib_s32 *scale);</pre>
DESCRIPTION	<p>Each of the functions in this group performs Inverse Fast Fourier Transform (IFFT).</p> <p>The following equation is used for forward FFT:</p> $dst[k] = \frac{1}{C1} \sum_{n=0}^{N-1} \{src[n] * \exp(-j2*PI*n*k/N)\}$ <p>and the following equation is used for inverse FFT (IFFT):</p> $dst[n] = \frac{1}{C2} \sum_{k=0}^{N-1} \{src[k] * \exp(j2*PI*n*k/N)\}$ <p>where</p> $k = 0, 1, \dots, (N - 1)$ $n = 0, 1, \dots, (N - 1)$ $N = 2**order$ <p>The signal FFT/IFFT functions can be categorized into four groups according to the ScaleMode in the function names in the following form:</p> <pre>mllib_Signal[FFT IFFT]_ScaleMode_OutType_InType_OpMode() mllib_Signal[FFT IFFT]_ScaleMode_DataType_OpMode()</pre> <p>The scaling factors C1 and C2 used in the equations are defined as follows:</p> <ul style="list-style-type: none"> ■ For ScaleMode = 1, C1 = 1 and C2 = 2**order. ■ For ScaleMode = 2, C1 = 2**order and C2 = 1. ■ For ScaleMode = 3, C1 = C2 = 2** (order/2) when order is even, or C1 = 2** ((order+1)/2) and C2 = 2** ((order-1)/2) when order is odd.

mllib_SignalIFFT_4(3MLIB)

- For ScaleMode = 4, $C1 = 2^{**}P$ and $C2 = 2^{**}Q$, where P and Q are adaptive scaling factors and are generated by the functions.

For functions with only real parts for the source signal, the imaginary parts are assumed to be all zero. For functions with only real parts for the destination signal, the imaginary parts are discarded. The functions with only one data type in their names perform the operation in place.

PARAMETERS

Each function takes some of the following arguments:

<i>dstr</i>	Destination signal array that contains the real parts.
<i>dsti</i>	Destination signal array that contains the imaginary parts.
<i>srcr</i>	Source signal array that contains the real parts.
<i>srci</i>	Source signal array that contains the imaginary parts.
<i>dstc</i>	Complex destination signal array. <i>dstc</i> [2*i] contains the real parts, and <i>dstc</i> [2*i+1] contains the imaginary parts.
<i>src</i>	Complex source signal array. <i>src</i> [2*i] contains the real parts, and <i>src</i> [2*i+1] contains the imaginary parts.
<i>srcdstr</i>	Source and destination signal array that contains the real parts.
<i>srcdsti</i>	Source and destination signal array that contains the imaginary parts.
<i>srcdstc</i>	Complex source and destination signal array. <i>srcdstc</i> [2*i] contains the real parts, and <i>srcdstc</i> [2*i+1] contains the imaginary parts.
<i>order</i>	Order of the transformation. The base-2 logarithm of the number of data samples.
<i>scale</i>	Adaptive scaling factor.

RETURN VALUES

The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

`mllib_SignalFFT_1(3MLIB)`, `mllib_SignalFFT_2(3MLIB)`,
`mllib_SignalFFT_3(3MLIB)`, `mllib_SignalFFT_4(3MLIB)`,
`mllib_SignalIFFT_1(3MLIB)`, `mllib_SignalIFFT_2(3MLIB)`,
`mllib_SignalIFFT_3(3MLIB)`, `attributes(5)`

NAME	mllib_SignalIFFTW_1, mllib_SignalIFFTW_1_S16_S16, mllib_SignalIFFTW_1_S16C_S16C, mllib_SignalIFFTW_1_S16_S16C, mllib_SignalIFFTW_1_S16, mllib_SignalIFFTW_1_S16C, mllib_SignalIFFTW_1_F32_F32, mllib_SignalIFFTW_1_F32C_F32C, mllib_SignalIFFTW_1_F32_F32C, mllib_SignalIFFTW_1_F32, mllib_SignalIFFTW_1_F32C – signal Inverse Fast Fourier Transform with windowing (IFFTW)
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalIFFTW_1_S16_S16(mllib_s16 *dstr, mllib_s16 *dsti, const mllib_s16 *srcr, const mllib_s16 *srci, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_1_S16C_S16C(mllib_s16 *dstc, const mllib_s16 *srcc, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_1_S16_S16C(mllib_s16 *dstr, const mllib_s16 *srcc, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_1_S16(mllib_s16 *srcdstr, mllib_s16 *srcdsti, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_1_S16C(mllib_s16 *srcdstc, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_1_F32_F32(mllib_f32 *dstr, mllib_f32 *dsti, const mllib_f32 *srcr, const mllib_f32 *srci, const mllib_f32 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_1_F32C_F32C(mllib_f32 *dstc, const mllib_f32 *srcc, const mllib_f32 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_1_F32_F32C(mllib_f32 *dstr, const mllib_f32 *srcc, const mllib_f32 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_1_F32(mllib_f32 *srcdstr, mllib_f32 *srcdsti, const mllib_f32 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_1_F32C(mllib_f32 *srcdstc, const mllib_f32 *window, mllib_s32 order);</pre>
DESCRIPTION	<p>Each of the functions in this group performs Inverse Fast Fourier Transform with windowing (IFFTW).</p> <p>The FFTW functions use the following equation:</p> $dst[k] = \frac{1}{C1} \sum_{n=0}^{N-1} \{src[n] * window[n] * exp(-j2*PI*n*k/N)\}$ <p>and the IFFTW functions use the following equation:</p> $dst[n] = \frac{1}{C2} \sum_{k=0}^{N-1} \{src[k] * window[k] * exp(j2*PI*n*k/N)\}$

mllib_SignalIFFTW_1(3MLIB)

where

```
k = 0, 1, ..., (N - 1)
n = 0, 1, ..., (N - 1)
N = 2**order
```

The signal FFTW/IFFTW functions can be categorized into four groups according to the `ScaleMode` in the function names in the following form:

```
mllib_Signal[FFTW|IFFTW]_ScaleMode_OutType_InType_OpMode()
mllib_Signal[FFTW|IFFTW]_ScaleMode_DataType_OpMode()
```

The scaling factors `C1` and `C2` used in the equations are defined as follows:

- For `ScaleMode = 1`, `C1 = 1` and `C2 = 2**order`.
- For `ScaleMode = 2`, `C1 = 2**order` and `C2 = 1`.
- For `ScaleMode = 3`, `C1 = C2 = 2** (order/2)` when `order` is even, or `C1 = 2** ((order+1)/2)` and `C2 = 2** ((order-1)/2)` when `order` is odd.
- For `ScaleMode = 4`, `C1 = 2**P` and `C2 = 2**Q`, where `P` and `Q` are adaptive scaling factors and are generated by the functions.

For functions with only real parts for the source signal, the imaginary parts are assumed to be all zero. For functions with only real parts for the destination signal, the imaginary parts are discarded. The functions with only one data type in their names perform the operation in place.

PARAMETERS

Each function takes some of the following arguments:

<i>dstr</i>	Destination signal array that contains the real parts.
<i>dsti</i>	Destination signal array that contains the imaginary parts.
<i>srcr</i>	Source signal array that contains the real parts.
<i>srci</i>	Source signal array that contains the imaginary parts.
<i>dstc</i>	Complex destination signal array. <code>dstc[2*i]</code> contains the real parts, and <code>dstc[2*i+1]</code> contains the imaginary parts.
<i>srcc</i>	Complex source signal array. <code>srcc[2*i]</code> contains the real parts, and <code>srcc[2*i+1]</code> contains the imaginary parts.
<i>srcdstr</i>	Source and destination signal array that contains the real parts.
<i>srcdsti</i>	Source and destination signal array that contains the imaginary parts.
<i>srcdstc</i>	Complex source and destination signal array. <code>srcdstc[2*i]</code> contains the real parts, and <code>srcdstc[2*i+1]</code> contains the imaginary parts.
<i>window</i>	Window coefficient array with <code>2**order</code> real elements. The window coefficients are in <code>Q15</code> format for the <code>S16</code> data type, or in <code>float</code> format for the <code>F32</code> data type.

`mlib_SignalIFFTW_1(3MLIB)`

order Order of the transformation. The base-2 logarithm of the number of data samples.

RETURN VALUES Each function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mlib_SignalFFTW_1(3MLIB)`, `mlib_SignalFFTW_2(3MLIB)`, `mlib_SignalFFTW_3(3MLIB)`, `mlib_SignalFFTW_4(3MLIB)`, `mlib_SignalIFFTW_2(3MLIB)`, `mlib_SignalIFFTW_3(3MLIB)`, `mlib_SignalIFFTW_4(3MLIB)`, `attributes(5)`

mllib_SignalIFFTW_2(3MLIB)

NAME	mllib_SignalIFFTW_2, mllib_SignalIFFTW_2_S16_S16_Mod, mllib_SignalIFFTW_2_S16C_S16C_Mod, mllib_SignalIFFTW_2_S16_S16C_Mod, mllib_SignalIFFTW_2_S16_Mod, mllib_SignalIFFTW_2_S16C_Mod, mllib_SignalIFFTW_2_F32_F32, mllib_SignalIFFTW_2_F32C_F32C, mllib_SignalIFFTW_2_F32_F32C, mllib_SignalIFFTW_2_F32, mllib_SignalIFFTW_2_F32C – signal Inverse Fast Fourier Transform with windowing (IFFTW)
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalIFFTW_2_S16_S16_Mod(mllib_s16 *dstr, mllib_s16 *dsti, const mllib_s16 *srcr, const mllib_s16 *srci, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_2_S16C_S16C_Mod(mllib_s16 *dstc, const mllib_s16 *srcc, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_2_S16_S16C_Mod(mllib_s16 *dstr, const mllib_s16 *srcc, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_2_S16_Mod(mllib_s16 *srcdstr, mllib_s16 *srcdsti, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_2_S16C_Mod(mllib_s16 *srcdstc, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_2_F32_F32(mllib_f32 *dstr, mllib_f32 *dsti, const mllib_f32 *srcr, const mllib_f32 *srci, const mllib_f32 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_2_F32C_F32C(mllib_f32 *dstc, const mllib_f32 *srcc, const mllib_f32 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_2_F32_F32C(mllib_f32 *dstr, const mllib_f32 *srcc, const mllib_f32 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_2_F32(mllib_f32 *srcdstr, mllib_f32 *srcdsti, const mllib_f32 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_2_F32C(mllib_f32 *srcdstc, const mllib_f32 *window, mllib_s32 order);</pre>
DESCRIPTION	<p>Each of the functions in this group performs Inverse Fast Fourier Transform with windowing (IFFTW).</p> <p>The FFTW functions use the following equation:</p> $\text{dst}[k] = \frac{1}{C1} \sum_{n=0}^{N-1} \{\text{src}[n] * \text{window}[n] * \exp(-j2*PI*n*k/N)\}$ <p>and the IFFTW functions use the following equation:</p> $\text{dst}[n] = \frac{1}{C2} \sum_{k=0}^{N-1} \{\text{src}[k] * \text{window}[k] * \exp(j2*PI*n*k/N)\}$

where

```
k = 0, 1, ..., (N - 1)
n = 0, 1, ..., (N - 1)
N = 2**order
```

The signal FFTW/IFFTW functions can be categorized into four groups according to the `ScaleMode` in the function names in the following form:

```
mllib_Signal[FFTW|IFFTW]_ScaleMode_OutType_InType_OpMode()
mllib_Signal[FFTW|IFFTW]_ScaleMode_DataType_OpMode()
```

The scaling factors `C1` and `C2` used in the equations are defined as follows:

- For `ScaleMode = 1`, $C1 = 1$ and $C2 = 2^{**}order$.
- For `ScaleMode = 2`, $C1 = 2^{**}order$ and $C2 = 1$.
- For `ScaleMode = 3`, $C1 = C2 = 2^{**}(order/2)$ when `order` is even, or $C1 = 2^{**}((order+1)/2)$ and $C2 = 2^{**}((order-1)/2)$ when `order` is odd.
- For `ScaleMode = 4`, $C1 = 2^{**}P$ and $C2 = 2^{**}Q$, where `P` and `Q` are adaptive scaling factors and are generated by the functions.

For functions with only real parts for the source signal, the imaginary parts are assumed to be all zero. For functions with only real parts for the destination signal, the imaginary parts are discarded. The functions with only one data type in their names perform the operation in place.

PARAMETERS

Each function takes some of the following arguments:

<i>dstr</i>	Destination signal array that contains the real parts.
<i>dsti</i>	Destination signal array that contains the imaginary parts.
<i>srcr</i>	Source signal array that contains the real parts.
<i>srci</i>	Source signal array that contains the imaginary parts.
<i>dstc</i>	Complex destination signal array. <code>dstc[2*i]</code> contains the real parts, and <code>dstc[2*i+1]</code> contains the imaginary parts.
<i>src</i>	Complex source signal array. <code>src[2*i]</code> contains the real parts, and <code>src[2*i+1]</code> contains the imaginary parts.
<i>srcdstr</i>	Source and destination signal array that contains the real parts.
<i>srcdsti</i>	Source and destination signal array that contains the imaginary parts.
<i>srcdstc</i>	Complex source and destination signal array. <code>srcdstc[2*i]</code> contains the real parts, and <code>srcdstc[2*i+1]</code> contains the imaginary parts.
<i>window</i>	Window coefficient array with $2^{**}order$ real elements. The window coefficients are in Q15 format for the S16 data type, or in float format for the F32 data type.

mllib_SignalIFFTW_2(3MLIB)

order Order of the transformation. The base-2 logarithm of the number of data samples.

RETURN VALUES Each function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalFFT_1(3MLIB)`, `mllib_SignalFFT_2(3MLIB)`,
`mllib_SignalFFT_3(3MLIB)`, `mllib_SignalFFT_4(3MLIB)`,
`mllib_SignalIFFTW_1(3MLIB)`, `mllib_SignalIFFTW_3(3MLIB)`,
`mllib_SignalIFFTW_4(3MLIB)`, `attributes(5)`

NAME	mllib_SignalIFFTW_3, mllib_SignalIFFTW_3_S16_S16_Mod, mllib_SignalIFFTW_3_S16C_S16C_Mod, mllib_SignalIFFTW_3_S16_S16C_Mod, mllib_SignalIFFTW_3_S16_Mod, mllib_SignalIFFTW_3_S16C_Mod, mllib_SignalIFFTW_3_F32_F32, mllib_SignalIFFTW_3_F32C_F32C, mllib_SignalIFFTW_3_F32_F32C, mllib_SignalIFFTW_3_F32, mllib_SignalIFFTW_3_F32C – signal Inverse Fast Fourier Transform with windowing (IFFTW)
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalIFFTW_3_S16_S16_Mod(mllib_s16 *dstr, mllib_s16 *dsti, const mllib_s16 *srcr, const mllib_s16 *srci, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_3_S16C_S16C_Mod(mllib_s16 *dstc, const mllib_s16 *srcc, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_3_S16_S16C_Mod(mllib_s16 *dstr, const mllib_s16 *srcc, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_3_S16_Mod(mllib_s16 *srcdstr, mllib_s16 *srcdsti, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_3_S16C_Mod(mllib_s16 *srcdstc, const mllib_s16 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_3_F32_F32(mllib_f32 *dstr, mllib_f32 *dsti, const mllib_f32 *srcr, const mllib_f32 *srci, const mllib_f32 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_3_F32C_F32C(mllib_f32 *dstc, const mllib_f32 *srcc, const mllib_f32 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_3_F32_F32C(mllib_f32 *dstr, const mllib_f32 *srcc, const mllib_f32 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_3_F32(mllib_f32 *srcdstr, mllib_f32 *srcdsti, const mllib_f32 *window, mllib_s32 order); mllib_status mllib_SignalIFFTW_3_F32C(mllib_f32 *srcdstc, const mllib_f32 *window, mllib_s32 order);</pre>
DESCRIPTION	<p>Each of the functions in this group performs Inverse Fast Fourier Transform with windowing (IFFTW).</p> <p>The FFTW functions use the following equation:</p> $\text{dst}[k] = \frac{1}{C1} \sum_{n=0}^{N-1} \{\text{src}[n] * \text{window}[n] * \exp(-j2*PI*n*k/N)\}$ <p>and the IFFTW functions use the following equation:</p> $\text{dst}[n] = \frac{1}{C2} \sum_{k=0}^{N-1} \{\text{src}[k] * \text{window}[k] * \exp(j2*PI*n*k/N)\}$

mllib_SignalIFFTW_3(3MLIB)

where

```
k = 0, 1, ..., (N - 1)
n = 0, 1, ..., (N - 1)
N = 2**order
```

The signal FFTW/IFFTW functions can be categorized into four groups according to the `ScaleMode` in the function names in the following form:

```
mllib_Signal[FFTW|IFFTW]_ScaleMode_OutType_InType_OpMode()
mllib_Signal[FFTW|IFFTW]_ScaleMode_DataType_OpMode()
```

The scaling factors `C1` and `C2` used in the equations are defined as follows:

- For `ScaleMode = 1`, $C1 = 1$ and $C2 = 2^{**}order$.
- For `ScaleMode = 2`, $C1 = 2^{**}order$ and $C2 = 1$.
- For `ScaleMode = 3`, $C1 = C2 = 2^{**}(order/2)$ when `order` is even, or $C1 = 2^{**}((order+1)/2)$ and $C2 = 2^{**}((order-1)/2)$ when `order` is odd.
- For `ScaleMode = 4`, $C1 = 2^{**}P$ and $C2 = 2^{**}Q$, where `P` and `Q` are adaptive scaling factors and are generated by the functions.

For functions with only real parts for the source signal, the imaginary parts are assumed to be all zero. For functions with only real parts for the destination signal, the imaginary parts are discarded. The functions with only one data type in their names perform the operation in place.

PARAMETERS

Each function takes some of the following arguments:

<i>dstr</i>	Destination signal array that contains the real parts.
<i>dsti</i>	Destination signal array that contains the imaginary parts.
<i>srcr</i>	Source signal array that contains the real parts.
<i>srci</i>	Source signal array that contains the imaginary parts.
<i>dstc</i>	Complex destination signal array. <code>dstc[2*i]</code> contains the real parts, and <code>dstc[2*i+1]</code> contains the imaginary parts.
<i>srcc</i>	Complex source signal array. <code>srcc[2*i]</code> contains the real parts, and <code>srcc[2*i+1]</code> contains the imaginary parts.
<i>srcdstr</i>	Source and destination signal array that contains the real parts.
<i>srcdsti</i>	Source and destination signal array that contains the imaginary parts.
<i>srcdstc</i>	Complex source and destination signal array. <code>srcdstc[2*i]</code> contains the real parts, and <code>srcdstc[2*i+1]</code> contains the imaginary parts.
<i>window</i>	Window coefficient array with $2^{**}order$ real elements. The window coefficients are in <code>Q15</code> format for the <code>S16</code> data type, or in <code>float</code> format for the <code>F32</code> data type.

`mllib_SignalIFFTW_3(3MLIB)`

order Order of the transformation. The base-2 logarithm of the number of data samples.

RETURN VALUES Each function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalFFTW_1(3MLIB)`, `mllib_SignalFFTW_2(3MLIB)`, `mllib_SignalFFTW_3(3MLIB)`, `mllib_SignalFFTW_4(3MLIB)`, `mllib_SignalIFFTW_1(3MLIB)`, `mllib_SignalIFFTW_2(3MLIB)`, `mllib_SignalIFFTW_4(3MLIB)`, `attributes(5)`

mllib_SignalIFFTW_4(3MLIB)

NAME	<code>mllib_SignalIFFTW_4</code> , <code>mllib_SignalIFFTW_4_S16_S16</code> , <code>mllib_SignalIFFTW_4_S16C_S16C</code> , <code>mllib_SignalIFFTW_4_S16_S16C</code> , <code>mllib_SignalIFFTW_4_S16</code> , <code>mllib_SignalIFFTW_4_S16C</code> – signal Inverse Fast Fourier Transform with windowing (IFFTW)
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalIFFTW_4_S16_S16(mllib_s16 *dst, mllib_s16 *dsti, const mllib_s16 *src, const mllib_s16 *srci, const mllib_s16 *window, mllib_s32 order, mllib_s32 *scale); mllib_status mllib_SignalIFFTW_4_S16C_S16C(mllib_s16 *dstc, const mllib_s16 *src, const mllib_s16 *window, mllib_s32 order, mllib_s32 *scale); mllib_status mllib_SignalIFFTW_4_S16_S16C(mllib_s16 *dst, const mllib_s16 *src, const mllib_s16 *window, mllib_s32 order, mllib_s32 *scale); mllib_status mllib_SignalIFFTW_4_S16(mllib_s16 *srcdst, mllib_s16 *srcdsti, const mllib_s16 *window, mllib_s32 order, mllib_s32 *scale); mllib_status mllib_SignalIFFTW_4_S16C(mllib_s16 *srcdstc, const mllib_s16 *window, mllib_s32 order, mllib_s32 *scale);</pre>
DESCRIPTION	<p>Each of the functions in this group performs Inverse Fast Fourier Transform with windowing (IFFTW).</p> <p>The FFTW functions use the following equation:</p> $\text{dst}[k] = \frac{1}{C1} \sum_{n=0}^{N-1} \{ \text{src}[n] * \text{window}[n] * \exp(-j2*\text{PI}*n*k/N) \}$ <p>and the IFFTW functions use the following equation:</p> $\text{dst}[n] = \frac{1}{C2} \sum_{k=0}^{N-1} \{ \text{src}[k] * \text{window}[k] * \exp(j2*\text{PI}*n*k/N) \}$ <p>where</p> $k = 0, 1, \dots, (N - 1)$ $n = 0, 1, \dots, (N - 1)$ $N = 2**\text{order}$ <p>The signal FFTW/IFFTW functions can be categorized into four groups according to the ScaleMode in the function names in the following form:</p> <pre>mllib_Signal[FFTW IFFTW]_ScaleMode_OutType_InType_OpMode() mllib_Signal[FFTW IFFTW]_ScaleMode_DataType_OpMode()</pre> <p>The scaling factors C1 and C2 used in the equations are defined as follows:</p> <ul style="list-style-type: none">■ For ScaleMode = 1, C1 = 1 and C2 = 2**order.■ For ScaleMode = 2, C1 = 2**order and C2 = 1.

- For ScaleMode = 3, $C1 = C2 = 2^{**}(\text{order}/2)$ when order is even, or $C1 = 2^{**}((\text{order}+1)/2)$ and $C2 = 2^{**}((\text{order}-1)/2)$ when order is odd.
- For ScaleMode = 4, $C1 = 2^{**}P$ and $C2 = 2^{**}Q$, where P and Q are adaptive scaling factors and are generated by the functions.

For functions with only real parts for the source signal, the imaginary parts are assumed to be all zero. For functions with only real parts for the destination signal, the imaginary parts are discarded. The functions with only one data type in their names perform the operation in place.

PARAMETERS Each function takes some of the following arguments:

<i>dstr</i>	Destination signal array that contains the real parts.
<i>dsti</i>	Destination signal array that contains the imaginary parts.
<i>srcr</i>	Source signal array that contains the real parts.
<i>srci</i>	Source signal array that contains the imaginary parts.
<i>dstc</i>	Complex destination signal array. <i>dstc</i> [2*i] contains the real parts, and <i>dstc</i> [2*i+1] contains the imaginary parts.
<i>srcc</i>	Complex source signal array. <i>srcc</i> [2*i] contains the real parts, and <i>srcc</i> [2*i+1] contains the imaginary parts.
<i>srcdstr</i>	Source and destination signal array that contains the real parts.
<i>srcdsti</i>	Source and destination signal array that contains the imaginary parts.
<i>srcdstc</i>	Complex source and destination signal array. <i>srcdstc</i> [2*i] contains the real parts, and <i>srcdstc</i> [2*i+1] contains the imaginary parts.
<i>window</i>	Window coefficient array with 2**order real elements. The window coefficients are in Q15 format for the S16 data type, or in float format for the F32 data type.
<i>order</i>	Order of the transformation. The base-2 logarithm of the number of data samples.
<i>scale</i>	Adaptive scaling factor.

RETURN VALUES Each function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

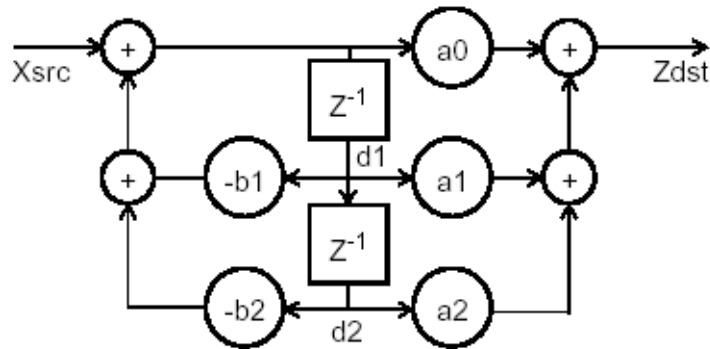
mllib_SignalIFFTW_4(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	MT-Safe

SEE ALSO mllib_SignalFFT_1(3MLIB), mllib_SignalFFT_2(3MLIB),
mllib_SignalFFT_3(3MLIB), mllib_SignalFFT_4(3MLIB),
mllib_SignalIFFTW_1(3MLIB), mllib_SignalIFFTW_2(3MLIB),
mllib_SignalIFFTW_3(3MLIB), attributes(5)

NAME	mllib_SignalIIR_Biquad_S16_S16_Sat, mllib_SignalIIR_Biquad_S16S_S16S_Sat, mllib_SignalIIR_Biquad_F32_F32, mllib_SignalIIR_Biquad_F32S_F32S – biquad Infinite Impulse Response (IIR) filtering
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalIIR_Biquad_S16_S16_Sat(mllib_s16 *dst, const mllib_s16 *src, void *filter, mllib_s32 n); mllib_status mllib_SignalIIR_Biquad_S16S_S16S_Sat(mllib_s16 *dst, const mllib_s16 *src, void *filter, mllib_s32 n); mllib_status mllib_SignalIIR_Biquad_F32_F32(mllib_f32 *dst, const mllib_f32 *src, void *filter, mllib_s32 n); mllib_status mllib_SignalIIR_Biquad_F32S_F32S(mllib_f32 *dst, const mllib_f32 *src, void *filter, mllib_s32 n);</pre>
DESCRIPTION	<p>Each of these functions applies a biquad IIR filter to a signal array.</p> $X = x(n) \quad n = 0, 1, \dots$ $Z = z(n) \quad n = 0, 1, \dots$ $= \sum_{k=0}^N a_k x(n-k) + \sum_{k=1}^M b_k z(n-k) \quad n = 0, 1, \dots$ $H(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}}$ <p>The biquad IIR filter is represented by the following figure:</p>

mllib_SignalIIR_Biquad_S16_S16_Sat(3MLIB)



PARAMETERS

Each of the functions takes the following arguments:

- dst* Destination signal array.
- src* Source signal array.
- filter* Internal filter structure.
- n* Number of samples in the source signal array.

RETURN VALUES

Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

- `mllib_SignalIIR_P4_S16_S16_Sat(3MLIB)`,
- `mllib_SignalIIRFree_Biquad_S16_S16(3MLIB)`,
- `mllib_SignalIIRFree_P4_S16_S16(3MLIB)`,
- `mllib_SignalIIRInit_Biquad_S16_S16(3MLIB)`,
- `mllib_SignalIIRInit_P4_S16_S16(3MLIB)`, `attributes(5)`

mllib_SignalIIRFree_Biquad_F32_F32(3MLIB)

NAME mllib_SignalIIRFree_Biquad_F32_F32, mllib_SignalIIRFree_Biquad_F32S_F32S – biquad Infinite Impulse Response (IIR) filtering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_SignalIIRFree_Biquad_F32_F32(void *filter);
void mllib_SignalIIRFree_Biquad_F32S_F32S(void *filter);
```

DESCRIPTION Each of these functions releases the memory allocated for the internal filter structure.

PARAMETERS Each of the functions takes the following arguments:
filter Internal filter structure.

RETURN VALUES None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalIIR_Biquad_S16_S16_Sat(3MLIB),
mllib_SignalIIR_P4_S16_S16_Sat(3MLIB),
mllib_SignalIIRFree_P4_S16_S16(3MLIB),
mllib_SignalIIRInit_Biquad_S16_S16(3MLIB),
mllib_SignalIIRInit_P4_S16_S16(3MLIB), attributes(5)

mllib_SignalIIRFree_Biquad_S16_S16(3MLIB)

NAME | mllib_SignalIIRFree_Biquad_S16_S16, mllib_SignalIIRFree_Biquad_S16S_S16S – biquad Infinite Impulse Response (IIR) filtering

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
void mllib_SignalIIRFree_Biquad_S16_S16(void *filter) ;  
void mllib_SignalIIRFree_Biquad_S16S_S16S(void *filter) ;
```

DESCRIPTION | Each of these functions releases the memory allocated for the internal filter structure.

PARAMETERS | Each of the functions takes the following arguments:
filter Internal filter structure.

RETURN VALUES | None.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_SignalIIR_Biquad_S16_S16_Sat(3MLIB),
mllib_SignalIIR_P4_S16_S16_Sat(3MLIB),
mllib_SignalIIRFree_P4_S16_S16(3MLIB),
mllib_SignalIIRInit_Biquad_S16_S16(3MLIB),
mllib_SignalIIRInit_P4_S16_S16(3MLIB), attributes(5)

mllib_SignalIIRFree_P4_F32_F32(3MLIB)

NAME mllib_SignalIIRFree_P4_F32_F32, mllib_SignalIIRFree_P4_F32S_F32S – parallel Infinite Impulse Response (IIR) filtering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_SignalIIRFree_P4_F32_F32(void *filter) ;
void mllib_SignalIIRFree_P4_F32S_F32S(void *filter) ;
```

DESCRIPTION Each of these functions releases the memory allocated for the internal filter structure.

PARAMETERS Each of the functions takes the following arguments:
filter Internal filter structure.

RETURN VALUES None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalIIR_Biquad_S16_S16_Sat(3MLIB),
mllib_SignalIIR_P4_S16_S16_Sat(3MLIB),
mllib_SignalIIRFree_Biquad_S16_S16(3MLIB),
mllib_SignalIIRInit_Biquad_S16_S16(3MLIB),
mllib_SignalIIRInit_P4_S16_S16(3MLIB), attributes(5)

mllib_SignalIIRFree_P4_S16_S16(3MLIB)

NAME | mllib_SignalIIRFree_P4_S16_S16, mllib_SignalIIRFree_P4_S16S_S16S – parallel Infinite Impulse Response (IIR) filtering

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
void mllib_SignalIIRFree_P4_S16_S16(void *filter) ;  
void mllib_SignalIIRFree_P4_S16S_S16S(void *filter) ;
```

DESCRIPTION | Each of these functions releases the memory allocated for the internal filter structure.

PARAMETERS | Each of the functions takes the following arguments:
filter Internal filter structure.

RETURN VALUES | None.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_SignalIIR_Biquad_S16_S16_Sat(3MLIB),
mllib_SignalIIR_P4_S16_S16_Sat(3MLIB),
mllib_SignalIIRFree_Biquad_S16_S16(3MLIB),
mllib_SignalIIRInit_Biquad_S16_S16(3MLIB),
mllib_SignalIIRInit_P4_S16_S16(3MLIB), attributes(5)

mlib_SignalIIRInit_Biquad_F32_F32(3MLIB)

NAME | mlib_SignalIIRInit_Biquad_F32_F32, mlib_SignalIIRInit_Biquad_F32S_F32S – biquad Infinite Impulse Response (IIR) filtering

SYNOPSIS |

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_SignalIIRInit_Biquad_F32_F32(void **filter, const
mlib_f32 *flt);

mlib_status mlib_SignalIIRInit_Biquad_F32S_F32S(void **filter,
const mlib_f32 *flt);
```

DESCRIPTION | Each of these functions allocates memory for the internal file structure and converts the filter coefficients into an internal representation.

PARAMETERS | Each of the functions takes the following arguments:
filter Internal filter structure.
flt Array of five filter coefficients: a0, a1, a2, b1, and b2.

RETURN VALUES | Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mlib_SignalIIR_Biquad_S16_S16_Sat(3MLIB),
mlib_SignalIIR_P4_S16_S16_Sat(3MLIB),
mlib_SignalIIRFree_Biquad_S16_S16(3MLIB),
mlib_SignalIIRFree_P4_S16_S16(3MLIB),
mlib_SignalIIRInit_P4_S16_S16(3MLIB), attributes(5)

mllib_SignalIIRInit_Biquad_S16_S16(3MLIB)

NAME | mllib_SignalIIRInit_Biquad_S16_S16, mllib_SignalIIRInit_Biquad_S16S_S16S – biquad Infinite Impulse Response (IIR) filtering

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalIIRInit_Biquad_S16_S16(void **filter, const
mllib_f32 *flt);

mllib_status mllib_SignalIIRInit_Biquad_S16S_S16S(void **filter,
const mllib_f32 *flt);
```

DESCRIPTION | Each of these functions allocates memory for the internal file structure and converts the filter coefficients into an internal representation.

PARAMETERS | Each of the functions takes the following arguments:

filter | Internal filter structure.

flt | Array of five filter coefficients: a0, a1, a2, b1, and b2.

RETURN VALUES | Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_SignalIIR_Biquad_S16_S16_Sat(3MLIB),
mllib_SignalIIR_P4_S16_S16_Sat(3MLIB),
mllib_SignalIIRFree_Biquad_S16_S16(3MLIB),
mllib_SignalIIRFree_P4_S16_S16(3MLIB),
mllib_SignalIIRInit_P4_S16_S16(3MLIB), attributes(5)

mlib_SignalIIRInit_P4_F32_F32(3MLIB)

NAME mlib_SignalIIRInit_P4_F32_F32, mlib_SignalIIRInit_P4_F32S_F32S – parallel Infinite Impulse Response (IIR) filtering

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_SignalIIRInit_P4_F32_F32(void **filter, const
mlib_f32 flt);

mlib_status mlib_SignalIIRInit_P4_F32S_F32S(void **filter, const
mlib_f32 flt);
```

DESCRIPTION Each of these functions allocates memory for the internal filter structure and converts the filter coefficients to an internal representation.

PARAMETERS Each of the functions takes the following arguments:

filter Internal filter structure.

flt Array of nine filter coefficients: c, a00, a10, b10, b20, a01, a11, b11, and b21.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mlib_SignalIIR_Biquad_S16_S16_Sat(3MLIB),
mlib_SignalIIR_P4_S16_S16_Sat(3MLIB),
mlib_SignalIIRFree_Biquad_S16_S16(3MLIB),
mlib_SignalIIRFree_P4_S16_S16(3MLIB),
mlib_SignalIIRInit_Biquad_S16_S16(3MLIB), attributes(5)

mllib_SignalIIRInit_P4_S16_S16(3MLIB)

NAME	mllib_SignalIIRInit_P4_S16_S16, mllib_SignalIIRInit_P4_S16S_S16S – parallel Infinite Impulse Response (IIR) filtering						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalIIRInit_P4_S16_S16(void **<i>filter</i>, const mllib_f32 <i>flt</i>); mllib_status mllib_SignalIIRInit_P4_S16S_S16S(void **<i>filter</i>, const mllib_f32 <i>flt</i>);</pre>						
DESCRIPTION	Each of these functions allocates memory for the internal filter structure and converts the filter coefficients to an internal representation.						
PARAMETERS	Each of the functions takes the following arguments: <i>filter</i> Internal filter structure. <i>flt</i> Array of nine filter coefficients: c, a00, a10, b10, b20, a01, a11, b11, and b21.						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_SignalIIR_Biquad_S16_S16_Sat(3MLIB), mllib_SignalIIR_P4_S16_S16_Sat(3MLIB), mllib_SignalIIRFree_Biquad_S16_S16(3MLIB), mllib_SignalIIRFree_P4_S16_S16(3MLIB), mllib_SignalIIRInit_Biquad_S16_S16(3MLIB), attributes(5)						

NAME mllib_SignalIIR_P4_S16_S16_Sat, mllib_SignalIIR_P4_S16S_S16S_Sat, mllib_SignalIIR_P4_F32_F32, mllib_SignalIIR_P4_F32S_F32S – parallel Infinite Impulse Response (IIR) filtering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalIIR_P4_S16_S16_Sat(mllib_s16 *dst, const
    mllib_s16 *src, void *filter, mllib_s32 n);

mllib_status mllib_SignalIIR_P4_S16S_S16S_Sat(mllib_s16 *dst, const
    mllib_s16 *src, void *filter, mllib_s32 n);

mllib_status mllib_SignalIIR_P4_F32_F32(mllib_f32 *dst, const
    mllib_f32 *src, void *filter, mllib_s32 n);

mllib_status mllib_SignalIIR_P4_F32S_F32S(mllib_f32 *dst, const
    mllib_f32 *src, void *filter, mllib_s32 n);
```

DESCRIPTION Each of these functions applies a fourth order parallel IIR filter to one signal packet and updates the filter state.

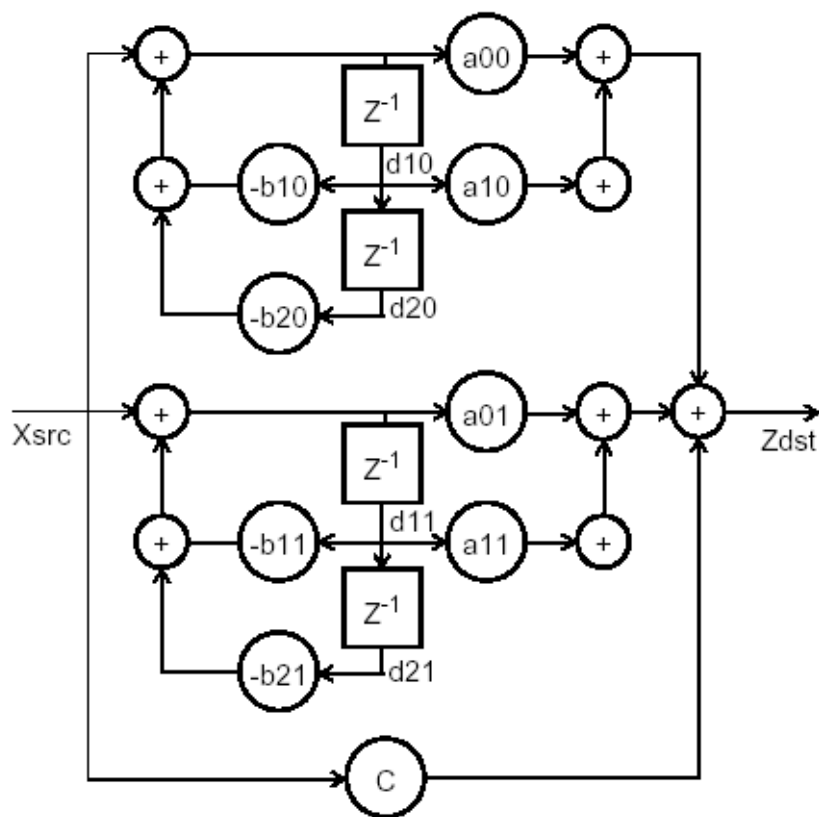
$$X = x(n) \quad n = 0, 1, \dots$$

$$Z = z(n) \quad n = 0, 1, \dots$$

$$= \sum_{k=0}^N a_k x(n-k) + \sum_{k=1}^M b_k z(n-k) \quad n = 0, 1, \dots$$

$$H(z) = C + \sum_{k=0}^1 \frac{(a_{0k} + a_{1k}z^{-1})}{(1 + b_{1k}z^{-1} + b_{2k}z^{-2})}$$

The fourth order parallel IIR filter is represented by the following figure:



PARAMETERS Each of the functions takes the following arguments:

- dst* Destination signal array.
- src* Source signal array.
- filter* Internal filter structure.
- n* Number of signal samples.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

mllib_SignalIIR_P4_S16_S16_Sat(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	MT-Safe

SEE ALSO

mllib_SignalIIR_Biquad_S16_S16_Sat(3MLIB),
mllib_SignalIIRFree_Biquad_S16_S16(3MLIB),
mllib_SignalIIRFree_P4_S16_S16(3MLIB),
mllib_SignalIIRInit_Biquad_S16_S16(3MLIB),
mllib_SignalIIRInit_P4_S16_S16(3MLIB), attributes(5)

mllib_SignalIMDCT_D64(3MLIB)

NAME | mllib_SignalIMDCT_D64 – Dolby AC-3 digital audio standard transformation

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
mllib_status mllib_SignalIMDCT_D64(mllib_d64 *data);
```

DESCRIPTION | The `mllib_SignalIMDCT_D64()` function performs the inverse modified discrete cosine transformation in Dolby's AC-3 digital audio standard.

PARAMETERS | The function takes the following arguments:

data | Pointer to the data array. `data[2*i]` contains the real parts, and `data[2*i+1]` contains the imaginary parts.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_SignalIMDCT_F32(3MLIB)`, `mllib_SignalIMDCTSplit_D64(3MLIB)`, `mllib_SignalIMDCTSplit_F32(3MLIB)`, `attributes(5)`

mllib_SignalIMDCT_F32(3MLIB)

NAME mllib_SignalIMDCT_F32 – Dolby AC-3 digital audio standard transformation

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalIMDCT_F32(mllib_f32 *data);
```

DESCRIPTION The mllib_SignalIMDCT_F32() function performs the inverse modified discrete cosine transformation in Dolby's AC-3 digital audio standard.

PARAMETERS The function takes the following arguments:

data Pointer to the data array. data[2*i] contains the real parts, and data[2*i+1] contains the imaginary parts.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_SignalIMDCT_D64\(3MLIB\)](#), [mllib_SignalIMDCTSplit_D64\(3MLIB\)](#), [mllib_SignalIMDCTSplit_F32\(3MLIB\)](#), [attributes\(5\)](#)

mllib_SignalIMDCTSplit_D64(3MLIB)

NAME mllib_SignalIMDCTSplit_D64 – Dolby AC-3 digital audio standard transformation

SYNOPSIS
cc [*flag...*] *file...* -lmllib [*library...*]
#include <mllib.h>

mllib_status **mllib_SignalIMDCTSplit_D64**(mllib_d64 **data*);

DESCRIPTION The mllib_SignalIMDCTSplit_D64() function performs the inverse modified discrete cosine transformation in Dolby's AC-3 digital audio standard.

PARAMETERS The function takes the following arguments:

data Pointer to the data array. data[4*i] contains the real parts of the first array, data[4*i+1] contains the real parts of the second array, data[4*i+2] contains the imaginary parts of the first array, and data[4*i+3] contains the imaginary parts of the second array.

RETURN VALUES The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalIMDCT_D64(3MLIB), mllib_SignalIMDCT_F32(3MLIB), mllib_SignalIMDCTSplit_F32(3MLIB), attributes(5)

mllib_SignalIMDCTSplit_F32(3MLIB)

- NAME** mllib_SignalIMDCTSplit_F32 – Dolby AC-3 digital audio standard transformation
- SYNOPSIS**

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
mllib_status mllib_SignalIMDCTSplit_F32(mllib_f32 *data);
```
- DESCRIPTION** The mllib_SignalIMDCTSplit_F32() function performs the inverse modified discrete cosine transformation in Dolby's AC-3 digital audio standard.
- PARAMETERS** The function takes the following arguments:
- data* Pointer to the data array. data[4*i] contains the real parts of the first array, data[4*i+1] contains the real parts of the second array, data[4*i+2] contains the imaginary parts of the first array, and data[4*i+3] contains the imaginary parts of the second array.
- RETURN VALUES** The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.
- ATTRIBUTES** See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

- SEE ALSO** mllib_SignalIMDCT_D64(3MLIB), mllib_SignalIMDCT_F32(3MLIB), mllib_SignalIMDCTSplit_D64(3MLIB), attributes(5)

mllib_SignalLimit(3MLIB)

NAME	<code>mllib_SignalLimit</code> , <code>mllib_SignalLimit_S16_S16</code> , <code>mllib_SignalLimit_S16S_S16S</code> , <code>mllib_SignalLimit_S16</code> , <code>mllib_SignalLimit_S16S</code> , <code>mllib_SignalLimit_F32_F32</code> , <code>mllib_SignalLimit_F32S_F32S</code> , <code>mllib_SignalLimit_F32</code> , <code>mllib_SignalLimit_F32S</code> – signal hard limiting
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalLimit_S16_S16(mllib_s16 *dst, const mllib_s16 *src, const mllib_s16 *low, const mllib_s16 *high, mllib_s32 n); mllib_status mllib_SignalLimit_S16S_S16S(mllib_s16 *dst, const mllib_s16 *src, const mllib_s16 *low, const mllib_s16 *high, mllib_s32 n); mllib_status mllib_SignalLimit_S16(mllib_s16 *srcdst, const mllib_s16 *low, const mllib_s16 *high, mllib_s32 n); mllib_status mllib_SignalLimit_S16S(mllib_s16 *srcdst, const mllib_s16 *low, const mllib_s16 *high, mllib_s32 n); mllib_status mllib_SignalLimit_F32_F32(mllib_f32 *dst, const mllib_f32 *src, const mllib_f32 *low, const mllib_f32 *high, mllib_s32 n); mllib_status mllib_SignalLimit_F32S_F32S(mllib_f32 *dst, const mllib_f32 *src, const mllib_f32 *low, const mllib_f32 *high, mllib_s32 n); mllib_status mllib_SignalLimit_F32(mllib_f32 *srcdst, const mllib_f32 *low, const mllib_f32 *high, mllib_s32 n); mllib_status mllib_SignalLimit_F32S(mllib_f32 *srcdst, const mllib_f32 *low, const mllib_f32 *high, mllib_s32 n);</pre>
DESCRIPTION	<p>Each of these functions performs hard limiting.</p> <p>For monaural signals, the following equation is used:</p> <pre>dst[i] = low[0] if src[i] < low[0] dst[i] = src[i] if low[0] ≤ src[i] < high[0] dst[i] = high[0] if src[i] ≥ high[0]</pre> <p>where $i = 0, 1, \dots, (n - 1)$.</p> <p>For stereo signals, the following equation is used:</p> <pre>dst[2*i] = low[0] if src[2*i] < low[0] dst[2*i] = src[2*i] if low[0] ≤ src[2*i] < high[0] dst[2*i] = high[0] if src[2*i] ≥ high[0] dst[2*i+1] = low[1] if src[2*i+1] < low[1] dst[2*i+1] = src[2*i+1] if low[1] ≤ src[2*i+1] < high[1] dst[2*i+1] = high[1] if src[2*i+1] ≥ high[1]</pre> <p>where $i = 0, 1, \dots, (n - 1)$.</p>

PARAMETERS Each of the functions takes some of the following arguments:

<i>dst</i>	Destination signal array.
<i>src</i>	Source signal array.
<i>srcdst</i>	Source and destination signal array.
<i>low</i>	Lower input limit. In the stereo version, <code>low[0]</code> contains the lower limit for channel 0, and <code>low[1]</code> contains the lower limit for channel 1.
<i>high</i>	Upper input limit. In the stereo version. <code>high[0]</code> contains the upper limit for channel 0, and <code>high[1]</code> contains the upper limit for channel 1.
<i>n</i>	Number of samples in the source signal array.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `attributes(5)`

mllib_SignalLinear2ADPCM2Bits(3MLIB)

NAME | mllib_SignalLinear2ADPCM2Bits – adaptive differential pulse code modulation (ADPCM)

SYNOPSIS |

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalLinear2ADPCM2Bits(mllib_u8 *adpcm, const
mllib_s16 *pcm, void *state, mllib_s32 n);
```

DESCRIPTION | The mllib_SignalLinear2ADPCM2Bits() function performs adaptive differential pulse code modulation (ADPCM) in compliance with the ITU (former CCITT) G.721, G.723, and G.726 specifications. It converts data from 2-bit ADPCM to 16-bit linear PCM to G.723 or G.726 16kbps format.

PARAMETERS | The function takes the following arguments:

adpcm ADPCM code array.

pcm Linear PCM sample array.

state Internal structure of the codec.

n Number of samples in the source array.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_SignalADPCM2Bits2Linear(3MLIB), mllib_SignalADPCM3Bits2Linear(3MLIB), mllib_SignalADPCM4Bits2Linear(3MLIB), mllib_SignalADPCM5Bits2Linear(3MLIB), mllib_SignalADPCMFree(3MLIB), mllib_SignalADPCMInit(3MLIB), mllib_SignalLinear2ADPCM3Bits(3MLIB), mllib_SignalLinear2ADPCM4Bits(3MLIB), mllib_SignalLinear2ADPCM5Bits(3MLIB), attributes(5)

NAME mllib_SignalLinear2ADPCM3Bits – adaptive differential pulse code modulation (ADPCM)

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalLinear2ADPCM3Bits(mllib_u8 *adpcm, const
    mllib_s16 *pcm, void *state, mllib_s32 n);
```

DESCRIPTION The `mllib_SignalLinear2ADPCM3Bits()` function performs adaptive differential pulse code modulation (ADPCM) in compliance with the ITU (former CCITT) G.721, G.723, and G.726 specifications. It converts data from 16-bit linear PCM to G.723 or G.726 24kbps 3-bit ADPCM format.

PARAMETERS The function takes the following arguments:

adpcm ADPCM code array.

pcm Linear PCM sample array.

state Internal structure of the codec.

n Number of samples in the source array.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalADPCM2Bits2Linear(3MLIB)`,
`mllib_SignalADPCM3Bits2Linear(3MLIB)`,
`mllib_SignalADPCM4Bits2Linear(3MLIB)`,
`mllib_SignalADPCM5Bits2Linear(3MLIB)`, `mllib_SignalADPCMFree(3MLIB)`,
`mllib_SignalADPCMInit(3MLIB)`, `mllib_SignalLinear2ADPCM2Bits(3MLIB)`,
`mllib_SignalLinear2ADPCM4Bits(3MLIB)`,
`mllib_SignalLinear2ADPCM5Bits(3MLIB)`, `attributes(5)`

mllib_SignalLinear2ADPCM4Bits(3MLIB)

NAME	mllib_SignalLinear2ADPCM4Bits – adaptive differential pulse code modulation (ADPCM)								
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmlib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalLinear2ADPCM4Bits(mllib_u8 *<i>adpcm</i>, const mllib_s16 *<i>pcm</i>, void *<i>state</i>, mllib_s32 <i>n</i>);</pre>								
DESCRIPTION	The <code>mllib_SignalLinear2ADPCM4Bits()</code> function performs adaptive differential pulse code modulation (ADPCM) in compliance with the ITU (former CCITT) G.721, G.723, and G.726 specifications. It converts data from 16-bit linear PCM to G.721 or G.726 32kbps 4-bit ADPCM format.								
PARAMETERS	The function takes the following arguments: <table><tr><td><i>adpcm</i></td><td>ADPCM code array.</td></tr><tr><td><i>pcm</i></td><td>Linear PCM sample array.</td></tr><tr><td><i>state</i></td><td>Internal structure of the codec.</td></tr><tr><td><i>n</i></td><td>Number of samples in the source array.</td></tr></table>	<i>adpcm</i>	ADPCM code array.	<i>pcm</i>	Linear PCM sample array.	<i>state</i>	Internal structure of the codec.	<i>n</i>	Number of samples in the source array.
<i>adpcm</i>	ADPCM code array.								
<i>pcm</i>	Linear PCM sample array.								
<i>state</i>	Internal structure of the codec.								
<i>n</i>	Number of samples in the source array.								
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .								
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe		
ATTRIBUTE TYPE	ATTRIBUTE VALUE								
Interface Stability	Evolving								
MT-Level	MT-Safe								
SEE ALSO	<code>mllib_SignalADPCM2Bits2Linear(3MLIB)</code> , <code>mllib_SignalADPCM3Bits2Linear(3MLIB)</code> , <code>mllib_SignalADPCM4Bits2Linear(3MLIB)</code> , <code>mllib_SignalADPCM5Bits2Linear(3MLIB)</code> , <code>mllib_SignalADPCMFree(3MLIB)</code> , <code>mllib_SignalADPCMInit(3MLIB)</code> , <code>mllib_SignalLinear2ADPCM2Bits(3MLIB)</code> , <code>mllib_SignalLinear2ADPCM3Bits(3MLIB)</code> , <code>mllib_SignalLinear2ADPCM5Bits(3MLIB)</code> , <code>attributes(5)</code>								

NAME mllib_SignalLinear2ADPCM5Bits – adaptive differential pulse code modulation (ADPCM)

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalLinear2ADPCM5Bits(mllib_u8 *adpcm, const
    mllib_s16 *pcm, void *state, mllib_s32 n);
```

DESCRIPTION The `mllib_SignalLinear2ADPCM5Bits()` function performs adaptive differential pulse code modulation (ADPCM) in compliance with the ITU (former CCITT) G.721, G.723, and G.726 specifications. It converts data from 16-bit linear PCM to G.723 or G.726 40kbps 5-bit ADPCM format.

PARAMETERS The function takes the following arguments:

adpcm ADPCM code array.

pcm Linear PCM sample array.

state Internal structure of the codec.

n Number of samples in the source array.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalADPCM2Bits2Linear(3MLIB)`,
`mllib_SignalADPCM3Bits2Linear(3MLIB)`,
`mllib_SignalADPCM4Bits2Linear(3MLIB)`,
`mllib_SignalADPCM5Bits2Linear(3MLIB)`, `mllib_SignalADPCMFree(3MLIB)`,
`mllib_SignalADPCMInit(3MLIB)`, `mllib_SignalLinear2ADPCM2Bits(3MLIB)`,
`mllib_SignalLinear2ADPCM3Bits(3MLIB)`,
`mllib_SignalLinear2ADPCM4Bits(3MLIB)`, `attributes(5)`

mllib_SignalLinear2ALaw(3MLIB)

NAME mllib_SignalLinear2ALaw – ITU G.711 m-law and A-law compression and decompression

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
mllib_status mllib_SignalLinear2ALaw(mllib_u8 *acode, const mllib_s16  
*pcm, mllib_s32 n);
```

DESCRIPTION The `mllib_SignalLinear2ALaw()` function performs ITU G.711 m-law and A-law compression and decompression in compliance with the ITU (Former CCITT) G.711 specification.

PARAMETERS The function takes the following arguments:

<i>acode</i>	A-law code array.
<i>pcm</i>	Linear PCM sample array.
<i>n</i>	Number of samples in the source array.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalALaw2Linear(3MLIB)`, `mllib_SignalALaw2uLaw(3MLIB)`, `mllib_SignalLinear2uLaw(3MLIB)`, `mllib_SignaluLaw2ALaw(3MLIB)`, `mllib_SignaluLaw2Linear(3MLIB)`, `attributes(5)`

mlib_SignalLinear2uLaw(3MLIB)

NAME mlib_SignalLinear2uLaw – ITU G.711 m-law and A-law compression and decompression

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_SignalLinear2uLaw(mlib_u8 *ucode, const mlib_s16
    *pcm, mlib_s32 n);
```

DESCRIPTION The `mlib_SignalLinear2uLaw()` function performs ITU G.711 m-law and A-law compression and decompression in compliance with the ITU (Former CCITT) G.711 specification.

PARAMETERS The function takes the following arguments:

ucode m-law code array.

pcm Linear PCM sample array.

n Number of samples in the source array.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mlib_SignalALaw2Linear(3MLIB)`, `mlib_SignalALaw2uLaw(3MLIB)`, `mlib_SignalLinear2ALaw(3MLIB)`, `mlib_SignaluLaw2ALaw(3MLIB)`, `mlib_SignaluLaw2Linear(3MLIB)`, `attributes(5)`

mllib_SignalLMSFilter_F32_F32(3MLIB)

NAME | mllib_SignalLMSFilter_F32_F32, mllib_SignalLMSFilter_F32S_F32S – least mean square (LMS) adaptive filtering

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalLMSFilter_F32_F32(mllib_f32 *dst, const
      mllib_f32 *src, const mllib_f32 *ref, void *filter, mllib_s32 n);

mllib_status mllib_SignalLMSFilter_F32S_F32S(mllib_f32 *dst, const
      mllib_f32 *src, const mllib_f32 *ref, void *filter, mllib_s32 n);
```

DESCRIPTION | Each of these functions applies the LMS adaptive filter on one signal packet and updates the filter states.

PARAMETERS | Each of the functions takes the following arguments:

<i>dst</i>	Destination signal array.
<i>src</i>	Source signal array.
<i>ref</i>	Referenced or "desired" signal array.
<i>filter</i>	Internal filter structure.
<i>n</i>	Number of samples in the source signal array.

RETURN VALUES | Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_SignalLMSFilterFree_S16_S16(3MLIB),
mllib_SignalLMSFilterInit_S16_S16(3MLIB), attributes(5)

mllib_SignalLMSFilterFree_F32_F32(3MLIB)

NAME mllib_SignalLMSFilterFree_F32_F32, mllib_SignalLMSFilterFree_F32S_F32S – least mean square (LMS) adaptive filtering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_SignalLMSFilterFree_F32_F32(void *filter);
void mllib_SignalLMSFilterFree_F32S_F32S(void *filter);
```

DESCRIPTION Each of these functions releases the memory allocated for the internal filter structure.

PARAMETERS Each of the functions takes the following arguments:
filter Internal filter structure.

RETURN VALUES None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalLMSFilter_S16_S16_Sat(3MLIB),
mllib_SignalLMSFilterInit_S16_S16(3MLIB), attributes(5)

mllib_SignalLMSFilterFree_S16_S16(3MLIB)

NAME mllib_SignalLMSFilterFree_S16_S16, mllib_SignalLMSFilterFree_S16S_S16S – least mean square (LMS) adaptive filtering

SYNOPSIS cc [*flag...*] *file...* -lmllib [*library...*]
#include <mllib.h>

```
void mllib_SignalLMSFilterFree_S16_S16(void *filter);  
void mllib_SignalLMSFilterFree_S16S_S16S(void *filter);
```

DESCRIPTION Each of these functions releases the memory allocated for the internal filter structure.

PARAMETERS Each of the functions takes the following arguments:

filter Internal filter structure.

RETURN VALUES None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalLMSFilter_S16_S16_Sat(3MLIB),
mllib_SignalLMSFilterInit_S16_S16(3MLIB), attributes(5)

mllib_SignalLMSFilterInit_F32_F32(3MLIB)

NAME mllib_SignalLMSFilterInit_F32_F32, mllib_SignalLMSFilterInit_F32S_F32S – least mean square (LMS) adaptive filtering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalLMSFilterInit_F32_F32(void **filter, const
    mllib_f32 *flt, mllib_s32 tap, mllib_f32 beta);

mllib_status mllib_SignalLMSFilterInit_F32S_F32S(void **filter, const
    mllib_f32 *flt, mllib_s32 tap, mllib_f32 beta);
```

DESCRIPTION Each of these functions allocates memory for the internal filter structure and converts the parameters into the internal representation.

PARAMETERS Each of the functions takes the following arguments:

filter Internal filter structure.

flt Filter coefficient array.

tap Taps of the filter.

beta Error weighting factor.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalLMSFilter_S16_S16_Sat(3MLIB),
mllib_SignalLMSFilterFree_S16_S16(3MLIB), attributes(5)

mllib_SignalLMSFilterInit_S16_S16(3MLIB)

NAME	mllib_SignalLMSFilterInit_S16_S16, mllib_SignalLMSFilterInit_S16S_S16S – least mean square (LMS) adaptive filtering								
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalLMSFilterInit_S16_S16(void **<i>filter</i>, const mllib_f32 *<i>flt</i>, mllib_s32 <i>tap</i>, mllib_f32 <i>beta</i>); mllib_status mllib_SignalLMSFilterInit_S16S_S16S(void **<i>filter</i>, const mllib_f32 *<i>flt</i>, mllib_s32 <i>tap</i>, mllib_f32 <i>beta</i>);</pre>								
DESCRIPTION	Each of these functions allocates memory for the internal filter structure and converts the parameters into the internal representation.								
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>filter</i></td><td>Internal filter structure.</td></tr><tr><td><i>flt</i></td><td>Filter coefficient array.</td></tr><tr><td><i>tap</i></td><td>Taps of the filter.</td></tr><tr><td><i>beta</i></td><td>Error weighting factor.</td></tr></table>	<i>filter</i>	Internal filter structure.	<i>flt</i>	Filter coefficient array.	<i>tap</i>	Taps of the filter.	<i>beta</i>	Error weighting factor.
<i>filter</i>	Internal filter structure.								
<i>flt</i>	Filter coefficient array.								
<i>tap</i>	Taps of the filter.								
<i>beta</i>	Error weighting factor.								
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.								
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe		
ATTRIBUTE TYPE	ATTRIBUTE VALUE								
Interface Stability	Evolving								
MT-Level	MT-Safe								
SEE ALSO	mllib_SignalLMSFilter_S16_S16_Sat(3MLIB), mllib_SignalLMSFilterFree_S16_S16(3MLIB), attributes(5)								

NAME mllib_SignalLMSFilter_S16_S16_Sat, mllib_SignalLMSFilter_S16S_S16S_Sat – least mean square (LMS) adaptive filtering

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalLMSFilter_S16_S16_Sat(mllib_s16 *dst, const
    mllib_s16 *src, const mllib_s16 *ref, void *filter, mllib_s32 n);

mllib_status mllib_SignalLMSFilter_S16S_S16S_Sat(mllib_s16 *dst,
    const mllib_s16 *src, const mllib_s16 *ref, void *filter, mllib_s32
    n);
```

DESCRIPTION Each of these functions applies the LMS adaptive filter on one signal packet and updates the filter states.

PARAMETERS Each of the functions takes the following arguments:

dst Destination signal array.
src Source signal array.
ref Referenced or "desired" signal array.
filter Internal filter structure.
n Number of samples in the source signal array.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalLMSFilterFree_S16_S16(3MLIB),
 mllib_SignalLMSFilterInit_S16_S16(3MLIB), attributes(5)

mllib_SignalLPC2Cepstral_F32(3MLIB)

NAME	mllib_SignalLPC2Cepstral_F32 – convert linear prediction coefficients to cepstral coefficients						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalLPC2Cepstral_F32(mllib_f32 *cepst, const mllib_f32 *lpc, mllib_f32 gain, mllib_s32 length, mllib_s32 order);</pre>						
DESCRIPTION	<p>The <code>mllib_SignalLPC2Cepstral_F32()</code> function converts linear prediction coefficients to cepstral coefficients.</p> <p>The cepstral coefficients are the coefficients of the Fourier transform representation of the log magnitude spectrum.</p> <p>The LPC cepstral coefficients can be derived recursively from the LPC coefficients as following.</p> $c(0) = \log(G)$ $c(m) = a(m) + \sum_{k=1}^{m-1} c(k) * a(m-k), \quad 1 \leq m \leq M$ $c(m) = \sum_{k=1}^{m-1} c(k) * a(m-k), \quad m > M$ <p>See <i>Fundamentals of Speech Recognition</i> by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>cepst</i> The cepstral coefficients.</p> <p><i>lpc</i> The linear prediction coefficients.</p> <p><i>gain</i> The gain of the LPC model.</p> <p><i>length</i> The length of the cepstral coefficients.</p> <p><i>order</i> The order of the linear prediction filter.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						

`mllib_SignalLPC2Cepstral_F32(3MLIB)`

SEE ALSO `mllib_SignalLPC2Cepstral_S16(3MLIB)`,
`mllib_SignalLPC2Cepstral_S16_Adp(3MLIB)`,
`mllib_SignalLPC2Cepstral_F32(3MLIB)`, `attributes(5)`

mllib_SignalLPC2Cepstral_S16(3MLIB)

NAME	mllib_SignalLPC2Cepstral_S16 – convert linear prediction coefficients to cepstral coefficients																
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalLPC2Cepstral_S16(mllib_s16 *cepst, mllib_s32 cscale, const mllib_s16 *lpc, mllib_s32 lscale, mllib_s16 gain, mllib_s32 gscale, mllib_s32 length, mllib_s32 order);</pre>																
DESCRIPTION	<p>The <code>mllib_SignalLPC2Cepstral_S16()</code> function converts linear prediction coefficients to cepstral coefficients. The user supplied scaling factor, <code>cscale</code>, will be used and the output will be saturated if necessary.</p> <p>The cepstral coefficients are the coefficients of the Fourier transform representation of the log magnitude spectrum.</p> <p>The LPC cepstral coefficients can be derived recursively from the LPC coefficients as following.</p> $c(0) = \log(G)$ $c(m) = a(m) + \sum_{k=1}^{m-1} c(k) * a(m-k), \quad 1 \leq m \leq M$ $c(m) = \sum_{k=1}^{m-1} c(k) * a(m-k), \quad m > M$ <p>See <i>Fundamentals of Speech Recognition</i> by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.</p>																
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><code>cepst</code></td><td>The cepstral coefficients.</td></tr><tr><td><code>cscale</code></td><td>The scaling factor of the cepstral coefficients, where <code>actual_data = output_data * 2**(-scaling_factor)</code>.</td></tr><tr><td><code>lpc</code></td><td>The linear prediction coefficients.</td></tr><tr><td><code>lscale</code></td><td>The scaling factor of the linear prediction coefficients, where <code>actual_data = input_data * 2**(-scaling_factor)</code>.</td></tr><tr><td><code>gain</code></td><td>The gain of the LPC model.</td></tr><tr><td><code>gscale</code></td><td>The scaling factor of the gain of the LPC model, where <code>actual_data = input_data * 2**(-scaling_factor)</code>.</td></tr><tr><td><code>length</code></td><td>The length of the cepstral coefficients.</td></tr><tr><td><code>order</code></td><td>The order of the linear prediction filter.</td></tr></table>	<code>cepst</code>	The cepstral coefficients.	<code>cscale</code>	The scaling factor of the cepstral coefficients, where <code>actual_data = output_data * 2**(-scaling_factor)</code> .	<code>lpc</code>	The linear prediction coefficients.	<code>lscale</code>	The scaling factor of the linear prediction coefficients, where <code>actual_data = input_data * 2**(-scaling_factor)</code> .	<code>gain</code>	The gain of the LPC model.	<code>gscale</code>	The scaling factor of the gain of the LPC model, where <code>actual_data = input_data * 2**(-scaling_factor)</code> .	<code>length</code>	The length of the cepstral coefficients.	<code>order</code>	The order of the linear prediction filter.
<code>cepst</code>	The cepstral coefficients.																
<code>cscale</code>	The scaling factor of the cepstral coefficients, where <code>actual_data = output_data * 2**(-scaling_factor)</code> .																
<code>lpc</code>	The linear prediction coefficients.																
<code>lscale</code>	The scaling factor of the linear prediction coefficients, where <code>actual_data = input_data * 2**(-scaling_factor)</code> .																
<code>gain</code>	The gain of the LPC model.																
<code>gscale</code>	The scaling factor of the gain of the LPC model, where <code>actual_data = input_data * 2**(-scaling_factor)</code> .																
<code>length</code>	The length of the cepstral coefficients.																
<code>order</code>	The order of the linear prediction filter.																
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .																

mllib_SignalLPC2Cepstral_S16(3MLIB)

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalLPC2Cepstral_S16(3MLIB),
mllib_SignalLPC2Cepstral_S16_Adp(3MLIB),
mllib_SignalLPC2Cepstral_F32(3MLIB), attributes(5)

mllib_SignalLPC2Cepstral_S16_Adp(3MLIB)

NAME	mllib_SignalLPC2Cepstral_S16_Adp – convert linear prediction coefficients to cepstral coefficients																
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalLPC2Cepstral_S16_Adp(mllib_s16 *cepst, mllib_s32 *cscale, const mllib_s16 *lpc, mllib_s32 lscale, mllib_s16 gain, mllib_s32 gscale, mllib_s32 length, mllib_s32 order);</pre>																
DESCRIPTION	<p>The <code>mllib_SignalLPC2Cepstral_S16_Adp()</code> function converts linear prediction coefficients to cepstral coefficients. The scaling factor of the output data, <code>cscale</code>, will be calculated based on the actual data.</p> <p>The cepstral coefficients are the coefficients of the Fourier transform representation of the log magnitude spectrum.</p> <p>The LPC cepstral coefficients can be derived recursively from the LPC coefficients as following.</p> $c(0) = \log(G)$ $c(m) = a(m) + \sum_{k=1}^{m-1} c(k) * a(m-k), \quad 1 \leq m \leq M$ $c(m) = \sum_{k=1}^{m-1} c(k) * a(m-k), \quad m > M$ <p>See <i>Fundamentals of Speech Recognition</i> by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.</p>																
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><code>cepst</code></td><td>The cepstral coefficients.</td></tr><tr><td><code>cscale</code></td><td>The scaling factor of the cepstral coefficients, where <code>actual_data = output_data * 2**(-scaling_factor)</code>.</td></tr><tr><td><code>lpc</code></td><td>The linear prediction coefficients.</td></tr><tr><td><code>lscale</code></td><td>The scaling factor of the linear prediction coefficients, where <code>actual_data = input_data * 2**(-scaling_factor)</code>.</td></tr><tr><td><code>gain</code></td><td>The gain of the LPC model.</td></tr><tr><td><code>gscale</code></td><td>The scaling factor of the gain of the LPC model, where <code>actual_data = input_data * 2**(-scaling_factor)</code>.</td></tr><tr><td><code>length</code></td><td>The length of the cepstral coefficients.</td></tr><tr><td><code>order</code></td><td>The order of the linear prediction filter.</td></tr></table>	<code>cepst</code>	The cepstral coefficients.	<code>cscale</code>	The scaling factor of the cepstral coefficients, where <code>actual_data = output_data * 2**(-scaling_factor)</code> .	<code>lpc</code>	The linear prediction coefficients.	<code>lscale</code>	The scaling factor of the linear prediction coefficients, where <code>actual_data = input_data * 2**(-scaling_factor)</code> .	<code>gain</code>	The gain of the LPC model.	<code>gscale</code>	The scaling factor of the gain of the LPC model, where <code>actual_data = input_data * 2**(-scaling_factor)</code> .	<code>length</code>	The length of the cepstral coefficients.	<code>order</code>	The order of the linear prediction filter.
<code>cepst</code>	The cepstral coefficients.																
<code>cscale</code>	The scaling factor of the cepstral coefficients, where <code>actual_data = output_data * 2**(-scaling_factor)</code> .																
<code>lpc</code>	The linear prediction coefficients.																
<code>lscale</code>	The scaling factor of the linear prediction coefficients, where <code>actual_data = input_data * 2**(-scaling_factor)</code> .																
<code>gain</code>	The gain of the LPC model.																
<code>gscale</code>	The scaling factor of the gain of the LPC model, where <code>actual_data = input_data * 2**(-scaling_factor)</code> .																
<code>length</code>	The length of the cepstral coefficients.																
<code>order</code>	The order of the linear prediction filter.																
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .																

mllib_SignalLPC2Cepstral_S16_Adp(3MLIB)

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalLPC2Cepstral_S16(3MLIB),
mllib_SignalLPC2Cepstral_S16_Adp(3MLIB),
mllib_SignalLPC2Cepstral_F32(3MLIB), attributes(5)

mllib_SignalLPC2LSP_F32(3MLIB)

NAME	mllib_SignalLPC2LSP_F32 – convert linear prediction coefficients to line spectral pair coefficients						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalLPC2LSP_F32(mllib_f32 *lsp, const mllib_f32 *lpc, mllib_s32 order);</pre>						
DESCRIPTION	<p>The <code>mllib_SignalLPC2LSP_F32()</code> function converts linear prediction coefficients to line spectral pair coefficients.</p> <p>The line spectral pair (LPS) coefficients are defined as the roots of the following two polynomials:</p> $P(z) = A(z) + z^{-(M+1)} * A(z^{-1})$ $Q(z) = A(z) - z^{-(M+1)} * A(z^{-1})$ <p>where $A(z)$ is the inverse filter</p> $A(z) = 1 - \sum_{i=1}^M a(i) * z^{-i}$ <p>Note that since $P(z)$ is symmetric and $Q(z)$ is antisymmetric all roots of these polynomials are on the unit circle and they alternate each other. $P(z)$ has a root at $z = -1$ ($w = \pi$) and $Q(z)$ has a root at $z = 1$ ($w = 0$).</p> <p>The line spectral frequency (LPF) are the angular frequency of the line spectral pair (LPS) coefficients.</p> $q = \cos(w)$ <p>where q is the LPS and w is the LPF.</p> <p>See <i>Fundamentals of Speech Recognition</i> by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>lsp</i></td><td>The line spectral pair coefficients.</td></tr><tr><td><i>lpc</i></td><td>The linear prediction coefficients.</td></tr><tr><td><i>order</i></td><td>The order of the linear prediction filter.</td></tr></table>	<i>lsp</i>	The line spectral pair coefficients.	<i>lpc</i>	The linear prediction coefficients.	<i>order</i>	The order of the linear prediction filter.
<i>lsp</i>	The line spectral pair coefficients.						
<i>lpc</i>	The linear prediction coefficients.						
<i>order</i>	The order of the linear prediction filter.						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						

`mlib_SignalLPC2LSP_F32(3MLIB)`

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mlib_SignalLSP2LPC_F32(3MLIB)`, `attributes(5)`

mllib_SignalLPC2LSP_S16(3MLIB)

NAME	mllib_SignalLPC2LSP_S16 – convert linear prediction coefficients to line spectral pair coefficients								
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalLPC2LSP_S16(mllib_s16 *lsp, const mllib_s16 *lpc, mllib_s32 lscale, mllib_s32 order);</pre>								
DESCRIPTION	<p>The <code>mllib_SignalLPC2LSP_S16()</code> function converts linear prediction coefficients to line spectral pair coefficients.</p> <p>The line spectral pair (LPS) coefficients are defined as the roots of the following two polynomials:</p> $P(z) = A(z) + z^{-(M+1)} * A(z^{-1})$ $Q(z) = A(z) - z^{-(M+1)} * A(z^{-1})$ <p>where $A(z)$ is the inverse filter</p> $A(z) = 1 - \sum_{i=1}^M a(i) * z^{-i}$ <p>Note that since $P(z)$ is symmetric and $Q(z)$ is antisymmetric all roots of these polynomials are on the unit circle and they alternate each other. $P(z)$ has a root at $z = -1$ ($w = \pi$) and $Q(z)$ has a root at $z = 1$ ($w = 0$).</p> <p>The line spectral frequency (LPF) are the angular frequency of the line spectral pair (LPS) coefficients.</p> $q = \cos(w)$ <p>where q is the LPS and w is the LPF.</p> <p>See <i>Fundamentals of Speech Recognition</i> by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.</p>								
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>lsp</i></td><td>The line spectral pair coefficients in Q15 format.</td></tr><tr><td><i>lpc</i></td><td>The linear prediction coefficients.</td></tr><tr><td><i>lscale</i></td><td>The scaling factor of the linear prediction coefficients, where <code>actual_data = input_data * 2**(-scaling_factor)</code>.</td></tr><tr><td><i>order</i></td><td>The order of the linear prediction filter.</td></tr></table>	<i>lsp</i>	The line spectral pair coefficients in Q15 format.	<i>lpc</i>	The linear prediction coefficients.	<i>lscale</i>	The scaling factor of the linear prediction coefficients, where <code>actual_data = input_data * 2**(-scaling_factor)</code> .	<i>order</i>	The order of the linear prediction filter.
<i>lsp</i>	The line spectral pair coefficients in Q15 format.								
<i>lpc</i>	The linear prediction coefficients.								
<i>lscale</i>	The scaling factor of the linear prediction coefficients, where <code>actual_data = input_data * 2**(-scaling_factor)</code> .								
<i>order</i>	The order of the linear prediction filter.								
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .								

`mlib_SignalLPC2LSP_S16(3MLIB)`

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mlib_SignalLSP2LPC_S16(3MLIB)`, `attributes(5)`

mllib_SignalLPCAutoCorrel_F32(3MLIB)

NAME	mllib_SignalLPCAutoCorrel_F32 – perform linear predictive coding with autocorrelation method						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalLPCAutoCorrel_F32(mllib_f32 *coeff, const mllib_f32 *signal, void *state);</pre>						
DESCRIPTION	<p>The <code>mllib_SignalLPCAutoCorrel_F32()</code> function performs linear predictive coding with autocorrelation method.</p> <p>In linear predictive coding (LPC) model, each speech sample is represented as a linear combination of the past M samples.</p> $s(n) = \sum_{i=1}^M a(i) * s(n-i) + G * u(n)$ <p>where $s(*)$ is the speech signal, $u(*)$ is the excitation signal, and G is the gain constants, M is the order of the linear prediction filter. Given $s(*)$, the goal is to find a set of coefficient $a(*)$ that minimizes the prediction error $e(*)$.</p> $e(n) = s(n) - \sum_{i=1}^M a(i) * s(n-i)$ <p>In autocorrelation method, the coefficients can be obtained by solving following set of linear equations.</p> $\sum_{i=1}^M a(i) * r(i-k) = r(k), \quad k=1, \dots, M$ <p>where</p> $r(k) = \sum_{j=0}^{N-k-1} s(j) * s(j+k)$ <p>are the autocorrelation coefficients of $s(*)$, N is the length of the input speech vector. $r(0)$ is the energy of the speech signal.</p> <p>Note that the autocorrelation matrix R is a Toeplitz matrix (symmetric with all diagonal elements equal), and the equations can be solved efficiently with Levinson-Durbin algorithm.</p> <p>See <i>Fundamentals of Speech Recognition</i> by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>coeff</i></td><td>The linear prediction coefficients.</td></tr><tr><td><i>signal</i></td><td>The input signal vector.</td></tr><tr><td><i>state</i></td><td>Pointer to the internal state structure.</td></tr></table>	<i>coeff</i>	The linear prediction coefficients.	<i>signal</i>	The input signal vector.	<i>state</i>	Pointer to the internal state structure.
<i>coeff</i>	The linear prediction coefficients.						
<i>signal</i>	The input signal vector.						
<i>state</i>	Pointer to the internal state structure.						

mllib_SignalLPCAutoCorrel_F32(3MLIB)

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalLPCAutoCorrelInit_F32(3MLIB)`,
`mllib_SignalLPCAutoCorrelGetEnergy_F32(3MLIB)`,
`mllib_SignalLPCAutoCorrelGetPARCOR_F32(3MLIB)`,
`mllib_SignalLPCAutoCorrelFree_F32(3MLIB)`, `attributes(5)`

mllib_SignalLPCAutoCorrelFree_S16(3MLIB)

NAME mllib_SignalLPCAutoCorrelFree_S16, mllib_SignalLPCAutoCorrelFree_F32 – clean up for autocorrelation method

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_SignalLPCAutoCorrelFree_S16(void *state);
void mllib_SignalLPCAutoCorrelFree_F32(void *state);
```

DESCRIPTION Each of these functions frees the internal state structure for autocorrelation method of linear predictive coding (LPC).
This function cleans up the internal state structure and releases all memory buffers.

PARAMETERS Each of the functions takes the following arguments:
state Pointer to the internal state structure.

RETURN VALUES None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalLPCAutoCorrelInit_S16(3MLIB),
mllib_SignalLPCAutoCorrelInit_F32(3MLIB),
mllib_SignalLPCAutoCorrel_S16(3MLIB),
mllib_SignalLPCAutoCorrel_S16_Adp(3MLIB),
mllib_SignalLPCAutoCorrel_F32(3MLIB),
mllib_SignalLPCAutoCorrelGetEnergy_S16(3MLIB),
mllib_SignalLPCAutoCorrelGetEnergy_F32(3MLIB),
mllib_SignalLPCAutoCorrelGetPARCOR_S16(3MLIB),
mllib_SignalLPCAutoCorrelGetPARCOR_F32(3MLIB), attributes(5)

mllib_SignalLPCAutoCorrelGetEnergy_F32(3MLIB)

NAME	mllib_SignalLPCAutoCorrelGetEnergy_F32 – return the energy of the input signal
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalLPCAutoCorrelGetEnergy_F32(mllib_f32 *energy, void *state);</pre>
DESCRIPTION	<p>The <code>mllib_SignalLPCAutoCorrelGetEnergy_F32()</code> function returns the energy of the input signal.</p> <p>In linear predictive coding (LPC) model, each speech sample is represented as a linear combination of the past M samples.</p> $s(n) = \sum_{i=1}^M a(i) * s(n-i) + G * u(n)$ <p>where $s(*)$ is the speech signal, $u(*)$ is the excitation signal, and G is the gain constants, M is the order of the linear prediction filter. Given $s(*)$, the goal is to find a set of coefficient $a(*)$ that minimizes the prediction error $e(*)$.</p> $e(n) = s(n) - \sum_{i=1}^M a(i) * s(n-i)$ <p>In autocorrelation method, the coefficients can be obtained by solving following set of linear equations.</p> $\sum_{i=1}^M a(i) * r(i-k) = r(k), \quad k=1, \dots, M$ <p>where</p> $r(k) = \sum_{j=0}^{N-k-1} s(j) * s(j+k)$ <p>are the autocorrelation coefficients of $s(*)$, N is the length of the input speech vector. $r(0)$ is the energy of the speech signal.</p> <p>Note that the autocorrelation matrix R is a Toeplitz matrix (symmetric with all diagonal elements equal), and the equations can be solved efficiently with Levinson-Durbin algorithm.</p> <p>See <i>Fundamentals of Speech Recognition</i> by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>energy</i> The energy of the input signal.</p> <p><i>state</i> Pointer to the internal state structure.</p>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .

mllib_SignalLPCAutoCorrelGetEnergy_F32(3MLIB)

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalLPCAutoCorrelInit_F32(3MLIB)`,
`mllib_SignalLPCAutoCorrel_F32(3MLIB)`,
`mllib_SignalLPCAutoCorrelGetPARCOR_F32(3MLIB)`,
`mllib_SignalLPCAutoCorrelFree_F32(3MLIB)`, `attributes(5)`

mllib_SignalLPCAutoCorrelGetEnergy_S16(3MLIB)

NAME	<code>mllib_SignalLPCAutoCorrelGetEnergy_S16</code> , <code>mllib_SignalLPCAutoCorrelGetEnergy_S16_Adp</code> – return the energy of the input signal
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalLPCAutoCorrelGetEnergy_S16(mllib_s16 *engery, mllib_s32 escale, void *state); mllib_status mllib_SignalLPCAutoCorrelGetEnergy_S16_Adp(mllib_s16 *engery, mllib_s32 *escale, void *state);</pre>
DESCRIPTION	<p>Each of the functions returns the energy of the input signal.</p> <p>In linear predictive coding (LPC) model, each speech sample is represented as a linear combination of the past M samples.</p> $s(n) = \sum_{i=1}^M a(i) * s(n-i) + G * u(n)$ <p>where $s(*)$ is the speech signal, $u(*)$ is the excitation signal, and G is the gain constants, M is the order of the linear prediction filter. Given $s(*)$, the goal is to find a set of coefficient $a(*)$ that minimizes the prediction error $e(*)$.</p> $e(n) = s(n) - \sum_{i=1}^M a(i) * s(n-i)$ <p>In autocorrelation method, the coefficients can be obtained by solving following set of linear equations.</p> $\sum_{i=1}^M a(i) * r(i-k) = r(k), \quad k=1, \dots, M$ <p>where</p> $r(k) = \sum_{j=0}^{N-k-1} s(j) * s(j+k)$ <p>are the autocorrelation coefficients of $s(*)$, N is the length of the input speech vector. $r(0)$ is the energy of the speech signal.</p> <p>Note that the autocorrelation matrix R is a Toeplitz matrix (symmetric with all diagonal elements equal), and the equations can be solved efficiently with Levinson-Durbin algorithm.</p> <p>See <i>Fundamentals of Speech Recognition</i> by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.</p>

mllib_SignalLPCAutoCorrelGetEnergy_S16(3MLIB)

Note for functions with adaptive scaling (with `_Adp` postfix), the scaling factor of the output data will be calculated based on the actual data; for functions with non-adaptive scaling (without `_Adp` postfix), the user supplied scaling factor will be used and the output will be saturated if necessary.

PARAMETERS Each function takes the following arguments:

energy The energy of the input signal.
escale The scaling factor of the energy, where `actual_data = output_data * 2**(-scaling_factor)`.
state Pointer to the internal state structure.

RETURN VALUES Each function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalLPCAutoCorrelInit_S16(3MLIB)`,
`mllib_SignalLPCAutoCorrel_S16(3MLIB)`,
`mllib_SignalLPCAutoCorrelGetPARCOR_S16(3MLIB)`,
`mllib_SignalLPCAutoCorrelFree_S16(3MLIB)`, `attributes(5)`

NAME	mllib_SignalLPCAutoCorrelGetPARCOR_F32 – return the partial correlation (PARCOR) coefficients
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalLPCAutoCorrelGetPARCOR_F32(mllib_f32 *parcor, void *state);</pre>
DESCRIPTION	<p>The <code>mllib_SignalLPCAutoCorrelGetPARCOR_F32()</code> function returns the partial correlation (PARCOR) coefficients.</p> <p>In linear predictive coding (LPC) model, each speech sample is represented as a linear combination of the past M samples.</p> $s(n) = \sum_{i=1}^M a(i) * s(n-i) + G * u(n)$ <p>where $s(*)$ is the speech signal, $u(*)$ is the excitation signal, and G is the gain constants, M is the order of the linear prediction filter. Given $s(*)$, the goal is to find a set of coefficient $a(*)$ that minimizes the prediction error $e(*)$.</p> $e(n) = s(n) - \sum_{i=1}^M a(i) * s(n-i)$ <p>In autocorrelation method, the coefficients can be obtained by solving following set of linear equations.</p> $\sum_{i=1}^M a(i) * r(i-k) = r(k), \quad k=1, \dots, M$ <p>where</p> $r(k) = \sum_{j=0}^{N-k-1} s(j) * s(j+k)$ <p>are the autocorrelation coefficients of $s(*)$, N is the length of the input speech vector. $r(0)$ is the energy of the speech signal.</p> <p>Note that the autocorrelation matrix R is a Toeplitz matrix (symmetric with all diagonal elements equal), and the equations can be solved efficiently with Levinson-Durbin algorithm.</p> <p>See <i>Fundamentals of Speech Recognition</i> by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>parcor</i> The partial correlation (PARCOR) coefficients.</p> <p><i>state</i> Pointer to the internal state structure.</p>

mllib_SignalLPCAutoCorrelGetPARCOR_F32(3MLIB)

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalLPCAutoCorrelInit_F32(3MLIB)`,
`mllib_SignalLPCAutoCorrel_F32(3MLIB)`,
`mllib_SignalLPCAutoCorrelGetEnergy_F32(3MLIB)`,
`mllib_SignalLPCAutoCorrelFree_F32(3MLIB)`, `attributes(5)`

mllib_SignalLPCAutoCorrelGetPARCOR_S16(3MLIB)

NAME	mllib_SignalLPCAutoCorrelGetPARCOR_S16, mllib_SignalLPCAutoCorrelGetPARCOR_S16_Adp – return the partial correlation (PARCOR) coefficients
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalLPCAutoCorrelGetPARCOR_S16(mllib_s16 *parcor, mllib_s32 pscale, void *state); mllib_status mllib_SignalLPCAutoCorrelGetPARCOR_S16_Adp(mllib_s16 *parcor, mllib_s32 *pscale, void *state);</pre>
DESCRIPTION	<p>Each of the functions returns the partial correlation (PARCOR) coefficients.</p> <p>In linear predictive coding (LPC) model, each speech sample is represented as a linear combination of the past M samples.</p> $s(n) = \sum_{i=1}^M a(i) * s(n-i) + G * u(n)$ <p>where $s(*)$ is the speech signal, $u(*)$ is the excitation signal, and G is the gain constants, M is the order of the linear prediction filter. Given $s(*)$, the goal is to find a set of coefficient $a(*)$ that minimizes the prediction error $e(*)$.</p> $e(n) = s(n) - \sum_{i=1}^M a(i) * s(n-i)$ <p>In autocorrelation method, the coefficients can be obtained by solving following set of linear equations.</p> $\sum_{i=1}^M a(i) * r(i-k) = r(k), \quad k=1, \dots, M$ <p>where</p> $r(k) = \sum_{j=0}^{N-k-1} s(j) * s(j+k)$ <p>are the autocorrelation coefficients of $s(*)$, N is the length of the input speech vector. $r(0)$ is the energy of the speech signal.</p> <p>Note that the autocorrelation matrix R is a Toeplitz matrix (symmetric with all diagonal elements equal), and the equations can be solved efficiently with Levinson-Durbin algorithm.</p> <p>See <i>Fundamentals of Speech Recognition</i> by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.</p>

mllib_SignalLPCAutoCorrelGetPARCOR_S16(3MLIB)

Note for functions with adaptive scaling (with `_Adp` postfix), the scaling factor of the output data will be calculated based on the actual data; for functions with non-adaptive scaling (without `_Adp` postfix), the user supplied scaling factor will be used and the output will be saturated if necessary.

PARAMETERS Each function takes the following arguments:

parcor The partial correlation (PARCOR) coefficients.
pscale The scaling factor of the partial correlation (PARCOR) coefficients, where $\text{actual_data} = \text{output_data} * 2^{**}(-\text{scaling_factor})$.
state Pointer to the internal state structure.

RETURN VALUES Each function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalLPCAutoCorrelInit_S16(3MLIB)`,
`mllib_SignalLPCAutoCorrel_S16(3MLIB)`,
`mllib_SignalLPCAutoCorrelGetEnergy_S16(3MLIB)`,
`mllib_SignalLPCAutoCorrelFree_S16(3MLIB)`, `attributes(5)`

mlib_SignalLPCAutoCorrelInit_S16(3MLIB)

NAME mlib_SignalLPCAutoCorrelInit_S16, mlib_SignalLPCAutoCorrelInit_F32 – initialization for autocorrelation method of linear predictive coding

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_SignalLPCAutoCorrelInit_S16(void *state, mlib_s32
length, mlib_s32 order);

mlib_status mlib_SignalLPCAutoCorrelInit_F32(void *state, mlib_s32
length, mlib_s32 order);
```

DESCRIPTION Each function initializes the internal state structure for autocorrelation method of linear predictive coding (LPC).

The init function performs internal state structure allocation and global initialization. Per LPC function call initialization is done in LPC function, so the same internal state structure can be reused for multiple LPC function calls.

PARAMETERS Each function takes the following arguments:

state Pointer to the internal state structure.

length The length of the input signal vector.

order The order of the linear prediction filter.

RETURN VALUES Each function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mlib_SignalLPCAutoCorrel_S16(3MLIB),
mlib_SignalLPCAutoCorrelGetEnergy_S16(3MLIB),
mlib_SignalLPCAutoCorrelGetPARCOR_S16(3MLIB),
mlib_SignalLPCAutoCorrelFree_S16(3MLIB), attributes(5)

mllib_SignalLPCAutoCorrel_S16(3MLIB)

NAME	mllib_SignalLPCAutoCorrel_S16, mllib_SignalLPCAutoCorrel_S16_Adp – perform linear predictive coding with autocorrelation method
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalLPCAutoCorrel_S16(mllib_s16 *coeff, mllib_s32 cscale, const mllib_s16 *signal, void *state); mllib_status mllib_SignalLPCAutoCorrel_S16_Adp(mllib_s16 *coeff, mllib_s32 *cscale, const mllib_s16 *signal, void *state);</pre>
DESCRIPTION	<p>Each function performs linear predictive coding with autocorrelation method.</p> <p>In linear predictive coding (LPC) model, each speech sample is represented as a linear combination of the past M samples.</p> $s(n) = \sum_{i=1}^M a(i) * s(n-i) + G * u(n)$ <p>where $s(*)$ is the speech signal, $u(*)$ is the excitation signal, and G is the gain constants, M is the order of the linear prediction filter. Given $s(*)$, the goal is to find a set of coefficient $a(*)$ that minimizes the prediction error $e(*)$.</p> $e(n) = s(n) - \sum_{i=1}^M a(i) * s(n-i)$ <p>In autocorrelation method, the coefficients can be obtained by solving following set of linear equations.</p> $\sum_{i=1}^M a(i) * r(i-k) = r(k), k=1, \dots, M$ <p>where</p> $r(k) = \sum_{j=0}^{N-k-1} s(j) * s(j+k)$ <p>are the autocorrelation coefficients of $s(*)$, N is the length of the input speech vector. $r(0)$ is the energy of the speech signal.</p> <p>Note that the autocorrelation matrix R is a Toeplitz matrix (symmetric with all diagonal elements equal), and the equations can be solved efficiently with Levinson-Durbin algorithm.</p> <p>See <i>Fundamentals of Speech Recognition</i> by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.</p> <p>Note for functions with adaptive scaling (with <code>_Adp</code> postfix), the scaling factor of the output data will be calculated based on the actual data; for functions with non-adaptive scaling (without <code>_Adp</code> postfix), the user supplied scaling factor will be used and the output will be saturated if necessary.</p>

PARAMETERS Each function takes the following arguments:

- coeff* The linear prediction coefficients.
- cscale* The scaling factor of the linear prediction coefficients, where
 $actual_data = output_data * 2^{(-scaling_factor)}$.
- signal* The input signal vector with samples in Q15 format.
- state* Pointer to the internal state structure.

RETURN VALUES Each function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_SignalLPCAutoCorrelInit_S16\(3MLIB\)](#),
[mllib_SignalLPCAutoCorrelGetEnergy_S16\(3MLIB\)](#),
[mllib_SignalLPCAutoCorrelGetPARCOR_S16\(3MLIB\)](#),
[mllib_SignalLPCAutoCorrelFree_S16\(3MLIB\)](#), [attributes\(5\)](#)

mllib_SignalLPCCovariance_F32(3MLIB)

NAME	mllib_SignalLPCCovariance_F32 – perform linear predictive coding with covariance method						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalLPCCovariance_F32(mllib_f32 *coeff, const mllib_f32 *signal, void *state);</pre>						
DESCRIPTION	<p>The <code>mllib_SignalLPCCovariance_F32()</code> function performs linear predictive coding with covariance method.</p> <p>In linear predictive coding (LPC) model, each speech sample is represented as a linear combination of the past M samples.</p> $s(n) = \sum_{i=1}^M a(i) * s(n-i) + G * u(n)$ <p>where $s(*)$ is the speech signal, $u(*)$ is the excitation signal, and G is the gain constants, M is the order of the linear prediction filter. Given $s(*)$, the goal is to find a set of coefficient $a(*)$ that minimizes the prediction error $e(*)$.</p> $e(n) = s(n) - \sum_{i=1}^M a(i) * s(n-i)$ <p>In covariance method, the coefficients can be obtained by solving following set of linear equations.</p> $\sum_{i=1}^M a(i) * c(i,k) = c(0,k), k=1, \dots, M$ <p>where</p> $c(i,k) = \sum_{j=0}^{N-k-1} s(j) * s(j+k-i)$ <p>are the covariance coefficients of $s(*)$, N is the length of the input speech vector.</p> <p>Note that the covariance matrix R is a symmetric matrix, and the equations can be solved efficiently with Cholesky decomposition method.</p> <p>See <i>Fundamentals of Speech Recognition</i> by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>coeff</i></td><td>The linear prediction coefficients.</td></tr><tr><td><i>signal</i></td><td>The input signal vector.</td></tr><tr><td><i>state</i></td><td>Pointer to the internal state structure.</td></tr></table>	<i>coeff</i>	The linear prediction coefficients.	<i>signal</i>	The input signal vector.	<i>state</i>	Pointer to the internal state structure.
<i>coeff</i>	The linear prediction coefficients.						
<i>signal</i>	The input signal vector.						
<i>state</i>	Pointer to the internal state structure.						

`mllib_SignalLPCcovariance_F32(3MLIB)`

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalLPCcovarianceInit_F32(3MLIB)`,
`mllib_SignalLPCcovarianceFree_F32(3MLIB)`, `attributes(5)`

mllib_SignalLPCCovarianceFree_S16(3MLIB)

NAME | mllib_SignalLPCCovarianceFree_S16, mllib_SignalLPCCovarianceFree_F32 – clean up for covariance method

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
void mllib_SignalLPCCovarianceFree_S16(void *state);  
void mllib_SignalLPCCovarianceFree_F32(void *state);
```

DESCRIPTION | Each of these functions frees the internal state structure for covariance method of linear predictive coding (LPC).

This function cleans up the internal state structure and releases all memory buffers.

PARAMETERS | Each of the functions takes the following arguments:

state Pointer to the internal state structure.

RETURN VALUES | None.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_SignalLPCCovarianceInit_S16(3MLIB),
mllib_SignalLPCCovarianceInit_F32(3MLIB),
mllib_SignalLPCCovariance_S16(3MLIB),
mllib_SignalLPCCovariance_S16_Adp(3MLIB),
mllib_SignalLPCCovariance_F32(3MLIB), attributes(5)

mlib_SignalLPCCovarianceInit_S16(3MLIB)

NAME mlib_SignalLPCCovarianceInit_S16, mlib_SignalLPCCovarianceInit_F32 – initialization for covariance method of linear predictive coding

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_SignalLPCCovarianceInit_S16(void *state, mlib_s32
length, mlib_s32 order);

mlib_status mlib_SignalLPCCovarianceInit_F32(void *state, mlib_s32
length, mlib_s32 order);
```

DESCRIPTION Each function initializes the internal state structure for covariance method of linear predictive coding (LPC).

The init function performs internal state structure allocation and global initialization. Per LPC function call initialization is done in LPC function, so the same internal state structure can be reused for multiple LPC function calls.

PARAMETERS Each function takes the following arguments:

state Pointer to the internal state structure.
length The length of the input signal vector.
order The order of the linear prediction filter.

RETURN VALUES Each function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mlib_SignalLPCCovariance_S16(3MLIB),
mlib_SignalLPCCovarianceFree_S16(3MLIB), attributes(5)

mllib_SignalLPCCovariance_S16(3MLIB)

NAME	mllib_SignalLPCCovariance_S16, mllib_SignalLPCCovariance_S16_Adp – perform linear predictive coding with covariance method
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalLPCCovariance_S16(mllib_s16 *coeff, mllib_s32 cscale, const mllib_s16 *signal, void *state); mllib_status mllib_SignalLPCCovariance_S16_Adp(mllib_s16 *coeff, mllib_s32 *cscale, const mllib_s16 *signal, void *state);</pre>
DESCRIPTION	<p>Each function performs linear predictive coding with covariance method.</p> <p>In linear predictive coding (LPC) model, each speech sample is represented as a linear combination of the past M samples.</p> $s(n) = \sum_{i=1}^M a(i) * s(n-i) + G * u(n)$ <p>where $s(*)$ is the speech signal, $u(*)$ is the excitation signal, and G is the gain constants, M is the order of the linear prediction filter. Given $s(*)$, the goal is to find a set of coefficient $a(*)$ that minimizes the prediction error $e(*)$.</p> $e(n) = s(n) - \sum_{i=1}^M a(i) * s(n-i)$ <p>In covariance method, the coefficients can be obtained by solving following set of linear equations.</p> $\sum_{i=1}^M a(i) * c(i,k) = c(0,k), \quad k=1, \dots, M$ <p>where</p> $c(i,k) = \sum_{j=0}^{N-k-1} s(j) * s(j+k-i)$ <p>are the covariance coefficients of $s(*)$, N is the length of the input speech vector.</p> <p>Note that the covariance matrix R is a symmetric matrix, and the equations can be solved efficiently with Cholesky decomposition method.</p> <p>See <i>Fundamentals of Speech Recognition</i> by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.</p> <p>Note for functions with adaptive scaling (with <code>_Adp</code> postfix), the scaling factor of the output data will be calculated based on the actual data; for functions with non-adaptive scaling (without <code>_Adp</code> postfix), the user supplied scaling factor will be used and the output will be saturated if necessary.</p>
PARAMETERS	Each function takes the following arguments:

`mllib_SignalLPCCovariance_S16(3MLIB)`

coeff The linear prediction coefficients.
cscale The scaling factor of the linear prediction coefficients, where $\text{actual_data} = \text{output_data} * 2^{**}(-\text{scaling_factor})$.
signal The input signal vector with samples in Q15 format.
state Pointer to the internal state structure.

RETURN VALUES Each function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalLPCCovarianceInit_S16(3MLIB)`,
`mllib_SignalLPCCovarianceFree_S16(3MLIB)`, `attributes(5)`

mllib_SignalLPCPerceptWeight_F32(3MLIB)

NAME	mllib_SignalLPCPerceptWeight_F32 – perform perceptual weighting on input signal						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalLPCPerceptWeight_F32(mllib_f32 *sigwgt, const mllib_f32 *signal, const mllib_f32 *lpc, mllib_f32 r1, mllib_f32 r2, void *state);</pre>						
DESCRIPTION	<p>The mllib_SignalLPCPerceptWeight_F32() function performs perceptual weighting on input signal.</p> <p>The perceptual weighting filter is defined as following.</p> $W(z) = \frac{A(z*r1)}{A(z*r2)}$ <p>where A(z) is the inverse filter</p> $A(z) = 1 - \sum_{i=1}^M a(i) * z^{-i}$ <p>See G.723.1, G.728, G.729, G.729A, GSM EFR standards.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>sigwgt</i> The weighted signal vector.</p> <p><i>signal</i> The input signal vector.</p> <p><i>lpc</i> The linear prediction coefficients.</p> <p><i>r1</i> The perceptual weighting filter coefficient, it is treated as 1 if 0 is supplied.</p> <p><i>r2</i> The perceptual weighting filter coefficient, it is treated as 1 if 0 is supplied.</p> <p><i>state</i> Pointer to the internal state structure.</p>						
RETURN VALUES	The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_SignalLPCPerceptWeightInit_F32(3MLIB), mllib_SignalLPCPerceptWeightFree_F32(3MLIB), attributes(5)						

mllib_SignalLPCPerceptWeightFree_S16(3MLIB)

- NAME** mllib_SignalLPCPerceptWeightFree_S16, mllib_SignalLPCPerceptWeightFree_F32 – clean up for perceptual weighting
- SYNOPSIS**

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_SignalLPCPerceptWeightFree_S16(void *state);
void mllib_SignalLPCPerceptWeightFree_F32(void *state);
```
- DESCRIPTION** Each of these functions frees the internal state structure for perceptual weighting of linear predictive coding (LPC).
This function cleans up the internal state structure and releases all memory buffers.
- PARAMETERS** Each of the functions takes the following arguments:
state Pointer to the internal state structure.
- RETURN VALUES** None.
- ATTRIBUTES** See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

- SEE ALSO** [mllib_SignalLPCPerceptWeightInit_S16\(3MLIB\)](#),
[mllib_SignalLPCPerceptWeightInit_F32\(3MLIB\)](#),
[mllib_SignalLPCPerceptWeight_S16\(3MLIB\)](#),
[mllib_SignalLPCPerceptWeight_F32\(3MLIB\)](#), [attributes\(5\)](#)

mllib_SignalLPCPerceptWeightInit_S16(3MLIB)

NAME	mllib_SignalLPCPerceptWeightInit_S16, mllib_SignalLPCPerceptWeightInit_F32 – initialization for perceptual weighting of linear predictive coding						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalLPCPerceptWeightInit_S16(void *state, mllib_s32 length, mllib_s32 order); mllib_status mllib_SignalLPCPerceptWeightInit_F32(void *state, mllib_s32 length, mllib_s32 order);</pre>						
DESCRIPTION	<p>Each function initializes the internal state structure for perceptual weighting of linear predictive coding (LPC).</p> <p>The init function performs internal state structure allocation and global initialization. Per LPC function call initialization is done in LPC function, so the same internal state structure can be reused for multiple LPC function calls.</p>						
PARAMETERS	<p>Each function takes the following arguments:</p> <p><i>state</i> Pointer to the internal state structure.</p> <p><i>length</i> The length of the input signal vector.</p> <p><i>order</i> The order of the linear prediction filter.</p>						
RETURN VALUES	Each function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_SignalLPCPerceptWeight_S16(3MLIB)</code> , <code>mllib_SignalLPCPerceptWeightFree_S16(3MLIB)</code> , <code>attributes(5)</code>						

mlib_SignalLPCPerceptWeight_S16(3MLIB)

NAME mlib_SignalLPCPerceptWeight_S16 – perform perceptual weighting on input signal

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_SignalLPCPerceptWeight_S16(mlib_s16 *sigwgt,
      const mlib_s16 *signal, const mlib_s16 *lpc, mlib_s32 lscale,
      mlib_s16 r1, mlib_s16 r2, void *state);
```

DESCRIPTION The mlib_SignalLPCPerceptWeight_S16() function performs perceptual weighting on input signal.

The perceptual weighting filter is defined as following.

$$W(z) = \frac{A(z*r1)}{A(z*r2)}$$

where A(z) is the inverse filter

$$A(z) = 1 - \sum_{i=1}^M a(i) * z^{-i}$$

See G.723.1, G.728, G.729, G.729A, GSM EFR standards.

PARAMETERS The function takes the following arguments:

sigwgt The weighted signal vector, the signal samples are in Q15 format.

signal The input signal vector, the signal samples are in Q15 format.

lpc The linear prediction coefficients.

lscale The scaling factor of the linear prediction coefficients, where actual_data = input_data * 2**(-scaling_factor).

r1 The perceptual weighting filter coefficient, the coefficient is in Q15 format, it is treated as 1 if 0 is supplied.

r2 The perceptual weighting filter coefficient, the coefficient is in Q15 format, it is treated as 1 if 0 is supplied.

state Pointer to the internal state structure.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

mllib_SignalLPCPerceptWeight_S16(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	MT-Safe

SEE ALSO [mllib_SignalLPCPerceptWeightInit_S16\(3MLIB\)](#),
[mllib_SignalLPCPerceptWeightFree_S16\(3MLIB\)](#), `attributes(5)`

NAME	mlib_SignalLPCPitchAnalyze_F32 – perform open-loop pitch analysis
SYNOPSIS	<pre>cc [flag...] file... -lmlib [library...] #include <mlib.h> mlib_status mlib_SignalLPCPitchAnalyze_F32(mlib_s32 *pitch, const mlib_f32 *sigwgt, const mlib_s32 *region, mlib_s32 length);</pre>
DESCRIPTION	<p>The <code>mlib_SignalLPCPitchAnalyze_F32()</code> function performs open-loop pitch analysis.</p> <p>The open-loop pitch analysis uses perceptual weighted signal and is done with following steps.</p> <p>In the first step, three maxima of the correlation</p> $R(k) = \sum_{j=0}^{N-1} sw(j) * sw(j-k)$ <p>where $N = \text{length}$, is located for each of the three search regions.</p> <p>In the second step, the retained maxima $R(T_i)$, $i=0, 1, 2$ are normalized as following.</p> $Rn(ti) = \frac{R(T_i)}{\sqrt{\sum_{j=0}^{N-1} sw(j-T_i)^2}}, \quad i=0, 1, 2$ <p>where $N = \text{length}$.</p> <p>In the third step, the best open-loop delay T_{opt} is determined as following.</p> <pre>Topt = T0 if (Rn(t1) >= (0.85 * Rn(Topt))) Topt = t1 if (Rn(t2) >= (0.85 * Rn(Topt))) Topt = t2</pre> <p>See G.729, G.729A, GSM EFR standards.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>pitch</i> The speech pitch estimated.</p> <p><i>sigwgt</i> The weighted signal vector. <i>sigwgt</i> points to the current sample of the weighted signal vector, <i>length</i> samples must be available after this point, and $\text{MAX}\{\text{region}[i], i=0, 1, \dots, 5\}$ samples must be available before this point.</p> <p><i>region</i> The lower/upper boundaries of the three search regions, where <i>region</i>[2*i] is the lower boundary of search region <i>i</i> and <i>region</i>[2*i+1] is the upper boundary of search region <i>i</i>.</p>

`mllib_SignalLPCPitchAnalyze_F32(3MLIB)`

length The length of the signal vectors over which the correlation is calculated.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalLPCPitchAnalyze_S16(3MLIB)`, `attributes(5)`

NAME	mllib_SignalLPCPitchAnalyze_S16 – perform open-loop pitch analysis
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalLPCPitchAnalyze_S16(mllib_s32 *pitch, const mllib_s16 *sigwgt, const mllib_s32 *region, mllib_s32 length);</pre>
DESCRIPTION	<p>The mllib_SignalLPCPitchAnalyze_S16() function performs open-loop pitch analysis.</p> <p>The open-loop pitch analysis uses perceptual weighted signal and is done with following steps.</p> <p>In the first step, three maxima of the correlation</p> $R(k) = \sum_{j=0}^{N-1} sw(j) * sw(j-k)$ <p>where $N = \text{length}$, is located for each of the three search regions.</p> <p>In the second step, the retained maxima $R(T_i)$, $i=0, 1, 2$ are normalized as following.</p> $Rn(ti) = \frac{R(T_i)}{\sqrt{\sum_{j=0}^{N-1} sw(j-T_i)^2}}, \quad i=0,1,2$ <p>where $N = \text{length}$.</p> <p>In the third step, the best open-loop delay T_{opt} is determined as following.</p> <pre>Topt = T0 if (Rn(t1) >= (0.85 * Rn(Topt))) Topt = t1 if (Rn(t2) >= (0.85 * Rn(Topt))) Topt = t2</pre> <p>See G.729, G.729A, GSM EFR standards.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>pitch</i> The speech pitch estimated.</p> <p><i>sigwgt</i> The weighted signal vector with samples in Q15 format. <i>sigwgt</i> points to the current sample of the weighted signal vector, <i>length</i> samples must be available after this point, and $\text{MAX}\{\text{region}[i], i=0, 1, \dots, 5\}$ samples must be available before this point.</p> <p><i>region</i> The lower/upper boundaries of the three search regions, where <i>region</i>[2*i] is the lower boundary of search region <i>i</i> and <i>region</i>[2*i+1] is the upper boundary of search region <i>i</i>.</p>

`mllib_SignalLPCPitchAnalyze_S16(3MLIB)`

length The length of the signal vectors over which the correlation is calculated.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalLPCPitchAnalyze_F32(3MLIB)`, `attributes(5)`

NAME	mllib_SignalLSP2LPC_F32 – convert line spectral pair coefficients to linear prediction coefficients
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalLSP2LPC_F32(mllib_f32 *lpc, const mllib_f32 *lsp, mllib_s32 order);</pre>
DESCRIPTION	<p>The mllib_SignalLSP2LPC_F32() function converts line spectral pair coefficients to linear prediction coefficients.</p> <p>The line spectral pair (LPS) coefficients are defined as the roots of the following two polynomials:</p> $P(z) = A(z) + z^{-(M+1)} * A(z^{-1})$ $Q(z) = A(z) - z^{-(M+1)} * A(z^{-1})$ <p>where $A(z)$ is the inverse filter</p> $A(z) = 1 - \sum_{i=1}^M a(i) * z^{-i}$ <p>Note that since $P(z)$ is symmetric and $Q(z)$ is antisymmetric all roots of these polynomials are on the unit circle and they alternate each other. $P(z)$ has a root at $z = -1$ ($w = \pi$) and $Q(z)$ has a root at $z = 1$ ($w = 0$).</p> <p>The line spectral frequency (LPF) are the angular frequency of the line spectral pair (LPS) coefficients.</p> $q = \cos(w)$ <p>where q is the LPS and w is the LPF.</p> <p>See <i>Fundamentals of Speech Recognition</i> by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>lpc</i> The linear prediction coefficients.</p> <p><i>lsp</i> The line spectral pair coefficients.</p> <p><i>order</i> The order of the linear prediction filter.</p>
RETURN VALUES	The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

mllib_SignalLSP2LPC_F32(3MLIB)

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_SignalLPC2LSP_F32\(3MLIB\)](#), `attributes(5)`

NAME	mllib_SignalLSP2LPC_S16, mllib_SignalLSP2LPC_S16_Adp – convert line spectral pair coefficients to linear prediction coefficients
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalLSP2LPC_S16(mllib_s16 *lpc, mllib_s32 lscale, const mllib_s16 *lsp, mllib_s32 order); mllib_status mllib_SignalLSP2LPC_S16_Adp(mllib_s16 *lpc, mllib_s32 *lscale, const mllib_s16 *lsp, mllib_s32 order);</pre>
DESCRIPTION	<p>Each of the functions in this group converts line spectral pair coefficients to linear prediction coefficients.</p> <p>The line spectral pair (LPS) coefficients are defined as the roots of the following two polynomials:</p> $P(z) = A(z) + z^{-(M+1)} * A(z^{-1})$ $Q(z) = A(z) - z^{-(M+1)} * A(z^{-1})$ <p>where $A(z)$ is the inverse filter</p> $A(z) = 1 - \sum_{i=1}^M a(i) * z^{-i}$ <p>Note that since $P(z)$ is symmetric and $Q(z)$ is antisymmetric all roots of these polynomials are on the unit circle and they alternate each other. $P(z)$ has a root at $z = -1$ ($w = \pi$) and $Q(z)$ has a root at $z = 1$ ($w = 0$).</p> <p>The line spectral frequency (LPS) are the angular frequency of the line spectral pair (LPS) coefficients.</p> $q = \cos(w)$ <p>where q is the LPS and w is the LPS.</p> <p>See <i>Fundamentals of Speech Recognition</i> by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.</p> <p>Note for functions with adaptive scaling (with <code>_Adp</code> postfix), the scaling factor of the output data will be calculated based on the actual data; for functions with non-adaptive scaling (without <code>_Adp</code> postfix), the user supplied scaling factor will be used and the output will be saturated if necessary.</p>
PARAMETERS	<p>Each function takes the following arguments:</p> <p><i>lpc</i> The linear prediction coefficients.</p> <p><i>lscale</i> The scaling factor of the line spectral pair coefficients, where <code>actual_data = output_data * 2**(-scaling_factor)</code>.</p>

`mllib_SignalLSP2LPC_S16(3MLIB)`

lsp The line spectral pair coefficients in Q15 format.
order The order of the linear prediction filter.

RETURN VALUES Each function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalLPC2LSP_S16(3MLIB)`, `attributes(5)`

NAME mllib_SignalMelCepstral_F32 – perform cepstral analysis in mel frequency scale

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMelCepstral_F32(mllib_f32 *cepst, const
      mllib_f32 *signal, void *state);
```

DESCRIPTION

The `mllib_SignalMelCepstral_F32()` function performs cepstral analysis in mel frequency scale.

The first two steps of mel scale cepstral analysis is the same as in general cepstral analysis. After the logarithm of the spectrum magnitude is obtained, it is converted into mel frequency scale before the inverse Fourier transform.

```

      +-----+           +-----+
      | Linear   |         | Inverse  |
      | to       |         | Fourier  |
...  ----->|----->|----->
      X'(k) | Mel Scale | X''(m) | Transform | c(n)
      +-----+           +-----+
```

where $X'(k)$ is defined in linear frequency scale and $X''(m)$ is defined in mel frequency scale.

The mel frequency scale is defined as following.

```
freq_mel = melmul * LOG10(1 + freq_linear / meldiv)
```

where `freq_mel` is the frequency in mel scale, `freq_linear` is the frequency in linear scale, `melmul` is the multiplying factor, `meldiv` is the dividing factor.

Optionally, a bank of band pass filters in linear frequency scale can be used below the bank of band pass filters in mel frequency scale, as shown below in linear frequency scale.

```
0  f1 f2 f3   fp fp+1 fp+2 fp+3 fp+q
|---|---|---| ... |---|---|---| ... | ... -> freq
```

where `fp = melbgn`, `fp+q = melend`, `p = nlinear`, `q = nmel`; the filters number 1 to `p` are defined in linear frequency scale which have equal bandwidth in linear frequency scale; the filters number `p+1` to `p+q` are defined in mel frequency scale which have equal bandwidth in mel frequency scale and increasing bandwidth in linear frequency scale.

See *Digital Signal Processing* by Alan V. Oppenheim and Ronald W. Schaffer, Prentice Hall, 1974.

See *Fundamentals of Speech Recognition* by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.

PARAMETERS

The function takes the following arguments:

`cepst` The cepstral coefficients.

`signal` The input signal vector.

`mllib_SignalMelCepstral_F32(3MLIB)`

state Pointer to the internal state structure.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalMelCepstralInit_F32(3MLIB)`,
`mllib_SignalMelCepstralFree_F32(3MLIB)`, `attributes(5)`

mllib_SignalMelCepstralFree_S16(3MLIB)

NAME mllib_SignalMelCepstralFree_S16, mllib_SignalMelCepstralFree_F32 – clean up for cepstral analysis in mel frequency scale

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_SignalMelCepstralFree_S16(void *state);
void mllib_SignalMelCepstralFree_F32(void *state);
```

DESCRIPTION Each of these functions frees the internal *state* structure and releases all memory buffers for cepstral analysis in mel frequency scale.

PARAMETERS Each of the functions takes the following arguments:
state Pointer to the internal state structure.

RETURN VALUES None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalMelCepstral_S16(3MLIB),
mllib_SignalMelCepstral_F32(3MLIB),
mllib_SignalMelCepstral_S16_Adp(3MLIB),
mllib_SignalMelCepstralInit_S16(3MLIB),
mllib_SignalMelCepstralInit_F32(3MLIB), attributes(5)

mllib_SignalMelCepstralInit_S16(3MLIB)

NAME	mllib_SignalMelCepstralInit_S16, mllib_SignalMelCepstralInit_F32 – initialization for cepstral analysis in mel frequency scale						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalMelCepstralInit_S16(void *<i>state</i>, mllib_s32 <i>nlinear</i>, mllib_s32 <i>nmel</i>, mllib_f32 <i>melbgn</i>, mllib_f32 <i>melend</i>, mllib_f32 <i>meldiv</i>, mllib_s32 <i>order</i>); mllib_status mllib_SignalMelCepstralInit_F32(void *<i>state</i>, mllib_s32 <i>nlinear</i>, mllib_s32 <i>nmel</i>, mllib_f32 <i>melbgn</i>, mllib_f32 <i>melend</i>, mllib_f32 <i>meldiv</i>, mllib_s32 <i>order</i>);</pre>						
DESCRIPTION	<p>Each of these functions initializes the internal state structure for cepstral analysis in mel frequency scale.</p> <p>The init function performs internal state structure allocation and global initialization. Per function call initialization is done in each function, so the same internal state structure can be reused for multiplefunction calls.</p>						
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <p><i>state</i> Pointer to the internal state structure.</p> <p><i>nlinear</i> The number of band pass filters in linear frequency scale.</p> <p><i>nmel</i> The number of band pass filters in mel frequency scale.</p> <p><i>melbgn</i> The begin radian frequency of the mel scale filter bank defined in linear frequency scale, where $0 \leq \text{melbgn} < \text{melend} \leq \text{PI}$, <i>melbgn</i> is ignored if <i>nlinear</i> = 0.</p> <p><i>melend</i> The end radian frequency of the mel scale filter bank defined in linear frequency scale, where $0 \leq \text{melbgn} < \text{melend} \leq \text{PI}$.</p> <p><i>meldiv</i> The dividing factor in linear to mel scale conversion, linear scale is measured in radians, with PI corresponding to half the sampling rate.</p> <p><i>order</i> The order of the input signal vector and the cepstral coefficients, where $\text{length} = 2**\text{order}$.</p>						
RETURN VALUES	Each of the functions returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						

mllib_SignalMelCepstralInit_S16(3MLIB)

SEE ALSO | mllib_SignalMelCepstral_S16(3MLIB),
mllib_SignalMelCepstral_F32(3MLIB),
mllib_SignalMelCepstral_S16_Adp(3MLIB),
mllib_SignalMelCepstralFree_S16(3MLIB),
mllib_SignalMelCepstralFree_F32(3MLIB), attributes(5)

mllib_SignalMelCepstral_S16(3MLIB)

NAME	mllib_SignalMelCepstral_S16 – perform cepstral analysis in mel frequency scale
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalMelCepstral_S16(mllib_s16 *cepst, mllib_s32 cscale, const mllib_s16 *signal, void *state);</pre>
DESCRIPTION	<p>The mllib_SignalMelCepstral_S16() function performs cepstral analysis in mel frequency scale. The user supplied scaling factor will be used and the output will be saturated if necessary.</p> <p>The first two steps of mel scale cepstral analysis is the same as in general cepstral analysis. After the logarithm of the spectrum magnitude is obtained, it is converted into mel frequency scale before the inverse Fourier transform.</p> <pre> +-----+ +-----+ Linear Inverse ... -----> to -----> Fourier -----> Mel Scale X'(m) Transform c(n) +-----+ +-----+</pre> <p>where $X'(k)$ is defined in linear frequency scale and $X'(m)$ is defined in mel frequency scale.</p> <p>The mel frequency scale is defined as following.</p> <pre>freq_mel = melmul * LOG10(1 + freq_linear / meldiv)</pre> <p>where <code>freq_mel</code> is the frequency in mel scale, <code>freq_linear</code> is the frequency in linear scale, <code>melmul</code> is the multiplying factor, <code>meldiv</code> is the dividing factor.</p> <p>Optionally, a bank of band pass filters in linear frequency scale can be used below the bank of band pass filters in mel frequency scale, as shown below in linear frequency scale.</p> <pre>0 f1 f2 f3 fp fp+1 fp+2 fp+3 fp+q --- --- --- ... --- --- --- -> freq</pre> <p>where <code>fp = melbgn</code>, <code>fp+q = melend</code>, <code>p = nlinear</code>, <code>q = nmel</code>; the filters number 1 to <code>p</code> are defined in linear frequency scale which have equal bandwidth in linear frequency scale; the filters number <code>p+1</code> to <code>p+q</code> are defined in mel frequency scale which have equal bandwidth in mel frequency scale and increasing bandwidth in linear frequency scale.</p> <p>See <i>Digital Signal Processing</i> by Alan V. Oppenheim and Ronald W. Schaffer, Prentice Hall, 1974.</p> <p>See <i>Fundamentals of Speech Recognition</i> by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.</p>
PARAMETERS	The function takes the following arguments: <code>cepst</code> The cepstral coefficients.

`mllib_SignalMelCepstral_S16(3MLIB)`

cscale The scaling factor of cepstral coefficients, where `actual_data = output_data * 2**(-scaling_factor)`.

signal The input signal vector, the signal samples are in Q15 format.

state Pointer to the internal state structure.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalMelCepstralInit_S16(3MLIB)`,
`mllib_SignalMelCepstral_S16_Adpt(3MLIB)`,
`mllib_SignalMelCepstralFree_S16(3MLIB)`, `attributes(5)`

mllib_SignalMelCepstral_S16_AdP(3MLIB)

NAME	mllib_SignalMelCepstral_S16_AdP – perform cepstral analysis in mel frequency scale
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalMelCepstral_S16_AdP(mllib_s16 *cepst, mllib_s32 *cscale, const mllib_s16 *signal, void *state);</pre>
DESCRIPTION	<p>The <code>mllib_SignalMelCepstral_S16_AdP()</code> function performs cepstral analysis in mel frequency scale. The scaling factor of the output data will be calculated based on the actual data.</p> <p>The first two steps of mel scale cepstral analysis is the same as in general cepstral analysis. After the logarithm of the spectrum magnitude is obtained, it is converted into mel frequency scale before the inverse Fourier transform.</p> <pre> +-----+ +-----+ Linear Inverse ... -----> to -----> Fourier -----> Mel Scale X'' (m) Transform c(n) +-----+ +-----+</pre> <p>where $X'(k)$ is defined in linear frequency scale and $X''(m)$ is defined in mel frequency scale.</p> <p>The mel frequency scale is defined as following.</p> <pre>freq_mel = melmul * LOG10(1 + freq_linear / meldiv)</pre> <p>where <code>freq_mel</code> is the frequency in mel scale, <code>freq_linear</code> is the frequency in linear scale, <code>melmul</code> is the multiplying factor, <code>meldiv</code> is the dividing factor.</p> <p>Optionally, a bank of band pass filters in linear frequency scale can be used below the bank of band pass filters in mel frequency scale, as shown below in linear frequency scale.</p> <pre>0 f1 f2 f3 fp fp+1 fp+2 fp+3 fp+q --- --- --- ... --- --- --- -> freq</pre> <p>where <code>fp = melbgn</code>, <code>fp+q = melend</code>, <code>p = nlinear</code>, <code>q = nmel</code>; the filters number 1 to <code>p</code> are defined in linear frequency scale which have equal bandwidth in linear frequency scale; the filters number <code>p+1</code> to <code>p+q</code> are defined in mel frequency scale which have equal bandwidth in mel frequency scale and increasing bandwidth in linear frequency scale.</p> <p>See <i>Digital Signal Processing</i> by Alan V. Oppenheim and Ronald W. Schaffer, Prentice Hall, 1974.</p> <p>See <i>Fundamentals of Speech Recognition</i> by Lawrence Rabiner and Biing-Hwang Juang, Prentice Hall, 1993.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <pre>cepst The cepstral coefficients.</pre>

`mllib_SignalMelCepstral_S16_AdP(3MLIB)`

scale The scaling factor of cepstral coefficients, where `actual_data = output_data * 2**(-scaling_factor)`.

signal The input signal vector, the signal samples are in Q15 format.

state Pointer to the internal state structure.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalMelCepstralInit_S16(3MLIB)`,
`mllib_SignalMelCepstral_S16(3MLIB)`,
`mllib_SignalMelCepstralFree_S16(3MLIB)`, `attributes(5)`

mllib_SignalMerge_F32S_F32(3MLIB)

NAME | mllib_SignalMerge_F32S_F32 – merge

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMerge_F32S_F32(mllib_f32 *dst, const
    mllib_f32 *ch0, const mllib_f32 *ch1, mllib_s32 n);
```

DESCRIPTION | The `mllib_SignalMerge_F32S_F32()` function merges two signal arrays to form a stereo signal array.

PARAMETERS | The function takes the following arguments:

<i>dst</i>	Output stereo signal array. <code>dst[2*i]</code> contains Channel 0, and <code>dst[2*i+1]</code> contains Channel 1.
<i>ch0</i>	Input signal array of Channel 0.
<i>ch1</i>	Input signal array of Channel 1.
<i>n</i>	Number of samples in the source signal arrays.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_SignalMerge_S16S_S16(3MLIB)`, `mllib_SignalSplit_F32_F32S(3MLIB)`, `mllib_SignalSplit_S16_S16S(3MLIB)`, `attributes(5)`

NAME mllib_SignalMerge_S16S_S16 – merge

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMerge_S16S_S16(mllib_s16 *dst, const
mllib_s16 *ch0, const mllib_s16 *ch1, mllib_s32 n);
```

DESCRIPTION The mllib_SignalMerge_S16S_S16() function merges two signal arrays to form a stereo signal array.

PARAMETERS The function takes the following arguments:

dst Output stereo signal array. *dst*[2**i*] contains Channel 0, and *dst*[2**i*+1] contains Channel 1.

ch0 Input signal array of Channel 0.

ch1 Input signal array of Channel 1.

n Number of samples in the source signal arrays.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalMerge_F32S_F32(3MLIB), mllib_SignalSplit_F32_F32S(3MLIB), mllib_SignalSplit_S16_S16S(3MLIB), attributes(5)

mllib_SignalMulBartlett_F32(3MLIB)

NAME mllib_SignalMulBartlett_F32, mllib_SignalMulBartlett_F32S – Bartlett windowing multiplication

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMulBartlett_F32(mllib_f32 *srcdst, mllib_s32
n);

mllib_status mllib_SignalMulBartlett_F32S(mllib_f32 *srcdst, mllib_s32
n);
```

DESCRIPTION Each of these functions performs multiplication of the Bartlett window.

PARAMETERS Each of the functions takes the following arguments:

srcdst Input and output signal array.

n Number of samples in signal and window arrays.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalMulBartlett_F32_F32(3MLIB), mllib_SignalMulBlackman_F32_F32(3MLIB), mllib_SignalMulBlackman_F32(3MLIB), mllib_SignalMulHamming_F32_F32(3MLIB), mllib_SignalMulHamming_F32(3MLIB), mllib_SignalMulHanning_F32_F32(3MLIB), mllib_SignalMulHanning_F32(3MLIB), mllib_SignalMulKaiser_F32_F32(3MLIB), mllib_SignalMulKaiser_F32(3MLIB), mllib_SignalMulRectangular_F32_F32(3MLIB), mllib_SignalMulRectangular_F32(3MLIB), mllib_SignalMulWindow_F32(3MLIB), mllib_SignalMulWindow_F32_F32(3MLIB), attributes(5)

mlib_SignalMulBartlett_F32_F32(3MLIB)

NAME | mlib_SignalMulBartlett_F32_F32, mlib_SignalMulBartlett_F32S_F32S – Bartlett windowing multiplication

SYNOPSIS |

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_SignalMulBartlett_F32_F32(mlib_f32 *dst, const
mlib_f32 *src, mlib_s32 n);

mlib_status mlib_SignalMulBartlett_F32S_F32S(mlib_f32 *dst, const
mlib_f32 *src, mlib_s32 n);
```

DESCRIPTION | Each of these functions performs multiplication of the Bartlett window.

PARAMETERS | Each of the functions takes the following arguments:

dst | Output signal array.

src | Input signal array.

n | Number of samples in signal and window arrays.

RETURN VALUES | Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mlib_SignalMulBartlett_F32(3MLIB),
mlib_SignalMulBlackman_F32_F32(3MLIB),
mlib_SignalMulBlackman_F32(3MLIB),
mlib_SignalMulHamming_F32_F32(3MLIB),
mlib_SignalMulHamming_F32(3MLIB),
mlib_SignalMulHanning_F32_F32(3MLIB),
mlib_SignalMulHanning_F32(3MLIB),
mlib_SignalMulKaiser_F32_F32(3MLIB),
mlib_SignalMulKaiser_F32(3MLIB),
mlib_SignalMulRectangular_F32_F32(3MLIB),
mlib_SignalMulRectangular_F32(3MLIB),
mlib_SignalMulWindow_F32(3MLIB),
mlib_SignalMulWindow_F32_F32(3MLIB), attributes(5)

mllib_SignalMulBartlett_S16(3MLIB)

NAME	mllib_SignalMulBartlett_S16, mllib_SignalMulBartlett_S16S – Bartlett windowing multiplication						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalMulBartlett_S16(mllib_s16 *srcdst, mllib_s32 n); mllib_status mllib_SignalMulBartlett_S16S(mllib_s16 *srcdst, mllib_s32 n);</pre>						
DESCRIPTION	Each of these functions performs multiplication of the Bartlett window.						
PARAMETERS	Each of the functions takes the following arguments: <i>srcdst</i> Input and output signal array. <i>n</i> Number of samples in signal and window arrays.						
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Interface Stability</td> <td style="text-align: center;">Evolving</td> </tr> <tr> <td style="text-align: center;">MT-Level</td> <td style="text-align: center;">MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<p>mllib_SignalMulBartlett_S16_S16(3MLIB), mllib_SignalMulBlackman_S16_S16(3MLIB), mllib_SignalMulBlackman_S16(3MLIB), mllib_SignalMulHamming_S16_S16(3MLIB), mllib_SignalMulHamming_S16(3MLIB), mllib_SignalMulHanning_S16_S16(3MLIB), mllib_SignalMulHanning_S16(3MLIB), mllib_SignalMulKaiser_S16_S16(3MLIB), mllib_SignalMulKaiser_S16(3MLIB), mllib_SignalMulRectangular_S16_S16(3MLIB), mllib_SignalMulRectangular_S16(3MLIB), mllib_SignalMulWindow_S16(3MLIB), mllib_SignalMulWindow_S16_S16(3MLIB), attributes(5)</p>						

mlib_SignalMulBartlett_S16_S16(3MLIB)

NAME | mlib_SignalMulBartlett_S16_S16, mlib_SignalMulBartlett_S16S_S16S – Bartlett windowing multiplication

SYNOPSIS |

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_SignalMulBartlett_S16_S16(mlib_s16 *dst, const
mlib_s16 *src, mlib_s32 n);

mlib_status mlib_SignalMulBartlett_S16S_S16S(mlib_s16 *dst, const
mlib_s16 *src, mlib_s32 n);
```

DESCRIPTION | Each of these functions performs multiplication of the Bartlett window.

PARAMETERS | Each of the functions takes the following arguments:

dst | Output signal array.

src | Input signal array.

n | Number of samples in signal and window arrays.

RETURN VALUES | Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mlib_SignalMulBartlett_S16(3MLIB),
mlib_SignalMulBlackman_S16(3MLIB),
mlib_SignalMulBlackman_S16_S16(3MLIB),
mlib_SignalMulHamming_S16(3MLIB),
mlib_SignalMulHamming_S16_S16(3MLIB),
mlib_SignalMulHanning_S16(3MLIB),
mlib_SignalMulHanning_S16_S16(3MLIB),
mlib_SignalMulKaiser_S16(3MLIB),
mlib_SignalMulKaiser_S16_S16(3MLIB),
mlib_SignalMulRectangular_S16(3MLIB),
mlib_SignalMulRectangular_S16_S16(3MLIB),
mlib_SignalMulWindow_S16_S16(3MLIB), attributes(5)

mllib_SignalMulBlackman_F32(3MLIB)

NAME	mllib_SignalMulBlackman_F32, mllib_SignalMulBlackman_F32S – Blackman windowing multiplication						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalMulBlackman_F32(mllib_f32 *<i>srcdst</i>, mllib_f32 <i>alpha</i>, mllib_s32 <i>n</i>); mllib_status mllib_SignalMulBlackman_F32S(mllib_f32 *<i>srcdst</i>, mllib_f32 <i>alpha</i>, mllib_s32 <i>n</i>);</pre>						
DESCRIPTION	Each of these functions performs multiplication of the Bartlett window.						
PARAMETERS	Each of the functions takes the following arguments: <i>srcdst</i> Input and output signal array. <i>alpha</i> Blackman window parameter. $-1 < \alpha < 0$. <i>n</i> Number of samples in signal and window arrays.						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1" data-bbox="446 1039 1412 1171"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_SignalMulBartlett_F32(3MLIB), mllib_SignalMulBartlett_F32_F32(3MLIB), mllib_SignalMulBlackman_F32_F32(3MLIB), mllib_SignalMulHamming_F32_F32(3MLIB), mllib_SignalMulHamming_F32(3MLIB), mllib_SignalMulHanning_F32_F32(3MLIB), mllib_SignalMulHanning_F32(3MLIB), mllib_SignalMulKaiser_F32_F32(3MLIB), mllib_SignalMulKaiser_F32(3MLIB), mllib_SignalMulRectangular_F32_F32(3MLIB), mllib_SignalMulRectangular_F32(3MLIB), mllib_SignalMulWindow_F32(3MLIB), mllib_SignalMulWindow_F32_F32(3MLIB), attributes(5)						

mllib_SignalMulBlackman_F32_F32(3MLIB)

NAME mllib_SignalMulBlackman_F32_F32, mllib_SignalMulBlackman_F32S_F32S – Blackman windowing multiplication

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMulBlackman_F32_F32(mllib_f32 *dst, const
mllib_f32 *src, mllib_f32 alpha, mllib_s32 n);

mllib_status mllib_SignalMulBlackman_F32S_F32S(mllib_f32 *dst, const
mllib_f32 *src, mllib_f32 alpha, mllib_s32 n);
```

DESCRIPTION Each of these functions performs multiplication of the Bartlett window.

PARAMETERS Each of the functions takes the following arguments:

dst Output signal array.

src Input signal array.

alpha Blackman window parameter. $-1 < \alpha < 0$.

n Number of samples in signal and window arrays.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalMulBartlett_F32(3MLIB),
mllib_SignalMulBartlett_F32_F32(3MLIB),
mllib_SignalMulBlackman_F32(3MLIB),
mllib_SignalMulHamming_F32_F32(3MLIB),
mllib_SignalMulHamming_F32(3MLIB),
mllib_SignalMulHanning_F32_F32(3MLIB),
mllib_SignalMulHanning_F32(3MLIB),
mllib_SignalMulKaiser_F32_F32(3MLIB),
mllib_SignalMulKaiser_F32(3MLIB),
mllib_SignalMulRectangular_F32_F32(3MLIB),
mllib_SignalMulRectangular_F32(3MLIB),
mllib_SignalMulWindow_F32(3MLIB),
mllib_SignalMulWindow_F32_F32(3MLIB), attributes(5)

mllib_SignalMulBlackman_S16(3MLIB)

NAME	mllib_SignalMulBlackman_S16, mllib_SignalMulBlackman_S16S – Blackman windowing multiplication						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalMulBlackman_S16(mllib_s16 *<i>srcdst</i>, mllib_f32 <i>alpha</i>, mllib_s32 <i>n</i>); mllib_status mllib_SignalMulBlackman_S16S(mllib_s16 *<i>srcdst</i>, mllib_f32 <i>alpha</i>, mllib_s32 <i>n</i>);</pre>						
DESCRIPTION	Each of these functions performs multiplication of the Bartlett window.						
PARAMETERS	Each of the functions takes the following arguments: <i>srcdst</i> Input and output signal array. <i>alpha</i> Blackman window parameter. $-1 < \alpha < 0$. <i>n</i> Number of samples in signal and window arrays.						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1" data-bbox="444 1039 1414 1171"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_SignalMulBartlett_S16_S16(3MLIB), mllib_SignalMulBartlett_S16(3MLIB), mllib_SignalMulBlackman_S16_S16(3MLIB), mllib_SignalMulHamming_S16(3MLIB), mllib_SignalMulHamming_S16_S16(3MLIB), mllib_SignalMulHanning_S16(3MLIB), mllib_SignalMulHanning_S16_S16(3MLIB), mllib_SignalMulKaiser_S16_S16(3MLIB), mllib_SignalMulKaiser_S16(3MLIB), mllib_SignalMulRectangular_S16_S16(3MLIB), mllib_SignalMulRectangular_S16(3MLIB), mllib_SignalMulWindow_S16(3MLIB), mllib_SignalMulWindow_S16_S16(3MLIB), attributes(5)						

mllib_SignalMulBlackman_S16_S16(3MLIB)

NAME mllib_SignalMulBlackman_S16_S16, mllib_SignalMulBlackman_S16S_S16S – Blackman windowing multiplication

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMulBlackman_S16_S16(mllib_s16 *dst, const
    mllib_s16 *src, mllib_f32 alpha, mllib_s32 n);

mllib_status mllib_SignalMulBlackman_S16S_S16S(mllib_s16 *dst, const
    mllib_s16 *src, mllib_f32 alpha, mllib_s32 n);
```

DESCRIPTION Each of these functions performs multiplication of the Bartlett window.

PARAMETERS Each of the functions takes the following arguments:

dst Output signal array.
src Input signal array.
alpha Blackman window parameter. $-1 < \alpha < 0$.
n Number of samples in signal and window arrays.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalMulBartlett_S16_S16(3MLIB),
mllib_SignalMulBartlett_S16(3MLIB),
mllib_SignalMulBlackman_S16(3MLIB),
mllib_SignalMulHamming_S16(3MLIB),
mllib_SignalMulHamming_S16_S16(3MLIB),
mllib_SignalMulHanning_S16(3MLIB),
mllib_SignalMulHanning_S16_S16(3MLIB),
mllib_SignalMulKaiser_S16_S16(3MLIB),
mllib_SignalMulKaiser_S16(3MLIB),
mllib_SignalMulRectangular_S16_S16(3MLIB),
mllib_SignalMulRectangular_S16(3MLIB),
mllib_SignalMulWindow_S16(3MLIB),
mllib_SignalMulWindow_S16_S16(3MLIB), attributes(5)

mllib_SignalMul_F32(3MLIB)

NAME | mllib_SignalMul_F32, mllib_SignalMul_F32S – multiplication

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMul_F32(mllib_f32 *src1dst, const mllib_f32
    *src2, mllib_s32 n);

mllib_status mllib_SignalMul_F32S(mllib_f32 *src1dst, const mllib_f32
    *src2, mllib_s32 n);
```

DESCRIPTION | Each of these functions performs multiplication.

PARAMETERS | Each of the functions takes the following arguments:

<i>src1dst</i>	The first input and the output signal array.
<i>src2</i>	The second input signal array.
<i>n</i>	Number of samples in the input signal arrays.

RETURN VALUES | Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `attributes(5)`

NAME mllib_SignalMul_F32_F32, mllib_SignalMul_F32S_F32S – multiplication

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMul_F32_F32(mllib_f32 *dst, const mllib_f32
    *src1, const mllib_f32 *src2, mllib_s32 n);

mllib_status mllib_SignalMul_F32S_F32S(mllib_f32 *dst, const
    mllib_f32 *src1, const mllib_f32 *src2, mllib_s32 n);
```

DESCRIPTION Each of these functions performs multiplication.

PARAMETERS Each of the functions takes the following arguments:

src1dst The output signal array.

src1 The first input signal array

src2 The second input signal array.

n Number of samples in the input signal arrays.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `attributes(5)`

mllib_SignalMulHamming_F32(3MLIB)

NAME mllib_SignalMulHamming_F32, mllib_SignalMulHamming_F32S – Bartlett windowing multiplication

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMulHamming_F32(mllib_f32 *srcdst, mllib_s32
n);

mllib_status mllib_SignalMulHamming_F32S(mllib_f32 *srcdst, mllib_s32
n);
```

DESCRIPTION Each of these functions performs multiplication of the Hamming window.

PARAMETERS Each of the functions takes the following arguments:

srcdst Input and output signal array.

n Number of samples in signal and window arrays.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalMulBartlett_F32(3MLIB),
mllib_SignalMulBartlett_F32_F32(3MLIB),
mllib_SignalMulBlackman_F32(3MLIB),
mllib_SignalMulBlackman_F32_F32(3MLIB),
mllib_SignalMulHamming_F32_F32(3MLIB),
mllib_SignalMulHanning_F32_F32(3MLIB),
mllib_SignalMulHanning_F32(3MLIB),
mllib_SignalMulKaiser_F32_F32(3MLIB),
mllib_SignalMulKaiser_F32(3MLIB),
mllib_SignalMulRectangular_F32_F32(3MLIB),
mllib_SignalMulRectangular_F32(3MLIB),
mllib_SignalMulWindow_F32(3MLIB),
mllib_SignalMulWindow_F32_F32(3MLIB), attributes(5)

mllib_SignalMulHamming_F32_F32(3MLIB)

NAME mllib_SignalMulHamming_F32_F32, mllib_SignalMulHamming_F32S_F32S – Hamming windowing multiplication

SYNOPSIS cc [*flag...*] *file...* -lmllib [*library...*]
#include <mllib.h>

```
mllib_status mllib_SignalMulHamming_F32_F32(mllib_f32 *dst, const
      mllib_f32 *src, mllib_s32 n);

mllib_status mllib_SignalMulHamming_F32S_F32S(mllib_f32 *dst, const
      mllib_f32 *src, mllib_s32 n);
```

DESCRIPTION Each of these functions performs multiplication of the Hamming window.

PARAMETERS Each of the functions takes the following arguments:

dst Output signal array.

src Input signal array.

n Number of samples in signal and window arrays.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalMulBartlett_F32(3MLIB),
mllib_SignalMulBartlett_F32_F32(3MLIB),
mllib_SignalMulBlackman_F32(3MLIB),
mllib_SignalMulBlackman_F32_F32(3MLIB),
mllib_SignalMulHamming_F32(3MLIB),
mllib_SignalMulHanning_F32_F32(3MLIB),
mllib_SignalMulHanning_F32(3MLIB),
mllib_SignalMulKaiser_F32_F32(3MLIB),
mllib_SignalMulKaiser_F32(3MLIB),
mllib_SignalMulRectangular_F32_F32(3MLIB),
mllib_SignalMulRectangular_F32(3MLIB),
mllib_SignalMulWindow_F32(3MLIB),
mllib_SignalMulWindow_F32_F32(3MLIB), attributes(5)

mllib_SignalMulHamming_S16(3MLIB)

NAME mllib_SignalMulHamming_S16, mllib_SignalMulHamming_S16S – Bartlett windowing multiplication

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
mllib_status mllib_SignalMulHamming_S16(mllib_s16 *srcdst, mllib_s32  
    n);  
  
mllib_status mllib_SignalMulHamming_S16S(mllib_s16 *srcdst, mllib_s32  
    n);
```

DESCRIPTION Each of these functions performs multiplication of the Hamming window.

PARAMETERS Each of the functions takes the following arguments:
srcdst Input and output signal array.
n Number of samples in signal and window arrays.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalMulBartlett_S16_S16(3MLIB),
mllib_SignalMulBartlett_S16(3MLIB),
mllib_SignalMulBlackman_S16_S16(3MLIB),
mllib_SignalMulBlackman_S16(3MLIB),
mllib_SignalMulHamming_S16_S16(3MLIB),
mllib_SignalMulHanning_S16_S16(3MLIB),
mllib_SignalMulHanning_S16(3MLIB),
mllib_SignalMulKaiser_S16_S16(3MLIB),
mllib_SignalMulKaiser_S16(3MLIB),
mllib_SignalMulRectangular_S16_S16(3MLIB),
mllib_SignalMulRectangular_S16(3MLIB),
mllib_SignalMulWindow_S16(3MLIB),
mllib_SignalMulWindow_S16_S16(3MLIB), attributes(5)

mlib_SignalMulHamming_S16_S16(3MLIB)

NAME mlib_SignalMulHamming_S16_S16, mlib_SignalMulHamming_S16S_S16S – Hamming windowing multiplication

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_SignalMulHamming_S16_S16(mlib_s16 *dst, const
mlib_s16 *src, mlib_s32 n);

mlib_status mlib_SignalMulHamming_S16S_S16S(mlib_s16 *dst, const
mlib_s16 *src, mlib_s32 n);
```

DESCRIPTION Each of these functions performs multiplication of the Hamming window.

PARAMETERS Each of the functions takes the following arguments:

dst Output signal array.

src Input signal array.

n Number of samples in signal and window arrays.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mlib_SignalMulBartlett_S16\(3MLIB\)](#),
[mlib_SignalMulBartlett_S16_S16\(3MLIB\)](#),
[mlib_SignalMulBlackman_S16\(3MLIB\)](#),
[mlib_SignalMulBlackman_S16_S16\(3MLIB\)](#),
[mlib_SignalMulHamming_S16\(3MLIB\)](#), [mlib_SignalMulHanning_S16\(3MLIB\)](#),
[mlib_SignalMulHanning_S16_S16\(3MLIB\)](#),
[mlib_SignalMulKaiser_S16\(3MLIB\)](#),
[mlib_SignalMulKaiser_S16_S16\(3MLIB\)](#),
[mlib_SignalMulRectangular_S16\(3MLIB\)](#),
[mlib_SignalMulRectangular_S16_S16\(3MLIB\)](#),
[mlib_SignalMulWindow_S16_S16\(3MLIB\)](#), [attributes\(5\)](#)

mllib_SignalMulHanning_F32(3MLIB)

NAME mllib_SignalMulHanning_F32, mllib_SignalMulHanning_F32S – Hanning windowing multiplication

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMulHanning_F32(mllib_f32 *srcdst, mllib_s32
n);

mllib_status mllib_SignalMulHanning_F32S(mllib_f32 *srcdst, mllib_s32
n);
```

DESCRIPTION Each of these functions performs multiplication of the Hanning window.

PARAMETERS Each of the functions takes the following arguments:

srcdst Source and destination signal array.

n Number of samples in signal and window arrays.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalMulBartlett_F32(3MLIB),
mllib_SignalMulBartlett_F32_F32(3MLIB),
mllib_SignalMulBlackman_F32(3MLIB),
mllib_SignalMulBlackman_F32_F32(3MLIB),
mllib_SignalMulHamming_F32(3MLIB),
mllib_SignalMulHamming_F32_F32(3MLIB),
mllib_SignalMulHanning_F32_F32(3MLIB),
mllib_SignalMulKaiser_F32_F32(3MLIB),
mllib_SignalMulKaiser_F32(3MLIB),
mllib_SignalMulRectangular_F32_F32(3MLIB),
mllib_SignalMulRectangular_F32(3MLIB),
mllib_SignalMulWindow_F32(3MLIB),
mllib_SignalMulWindow_F32_F32(3MLIB), attributes(5)

mllib_SignalMulHanning_F32_F32(3MLIB)

NAME mllib_SignalMulHanning_F32_F32, mllib_SignalMulHanning_F32S_F32S – Hanning windowing multiplication

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMulHanning_F32_F32(mllib_f32 *dst, const
mllib_f32 *src, mllib_s32 n);

mllib_status mllib_SignalMulHanning_F32S_F32S(mllib_f32 *dst, const
mllib_f32 *src, mllib_s32 n);
```

DESCRIPTION Each of these functions performs multiplication of the Hanning window.

PARAMETERS Each of the functions takes the following arguments:

dst Destination signal array.

src Source signal array.

n Number of samples in signal and window arrays.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalMulBartlett_F32(3MLIB),
mllib_SignalMulBartlett_F32_F32(3MLIB),
mllib_SignalMulBlackman_F32(3MLIB),
mllib_SignalMulBlackman_F32_F32(3MLIB),
mllib_SignalMulHamming_F32(3MLIB),
mllib_SignalMulHamming_F32_F32(3MLIB),
mllib_SignalMulHanning_F32(3MLIB),
mllib_SignalMulKaiser_F32_F32(3MLIB),
mllib_SignalMulKaiser_F32(3MLIB),
mllib_SignalMulRectangular_F32_F32(3MLIB),
mllib_SignalMulRectangular_F32(3MLIB),
mllib_SignalMulWindow_F32(3MLIB),
mllib_SignalMulWindow_F32_F32(3MLIB), attributes(5)

mllib_SignalMulHanning_S16(3MLIB)

NAME	mllib_SignalMulHanning_S16, mllib_SignalMulHanning_S16S – Hanning windowing multiplication
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalMulHanning_S16(mllib_s16 *srcdst, mllib_s32 n); mllib_status mllib_SignalMulHanning_S16S(mllib_s16 *srcdst, mllib_s32 n);</pre>
DESCRIPTION	Each of these functions performs multiplication of the Hanning window.
PARAMETERS	Each of the functions takes the following arguments: <i>srcdst</i> Source and destination signal array. <i>n</i> Number of samples in signal and window arrays.
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalMulBartlett_S16_S16(3MLIB),
mllib_SignalMulBartlett_S16(3MLIB),
mllib_SignalMulBlackman_S16_S16(3MLIB),
mllib_SignalMulBlackman_S16(3MLIB),
mllib_SignalMulHamming_S16_S16(3MLIB),
mllib_SignalMulHamming_S16(3MLIB),
mllib_SignalMulHanning_S16_S16(3MLIB),
mllib_SignalMulKaiser_S16_S16(3MLIB),
mllib_SignalMulKaiser_S16(3MLIB),
mllib_SignalMulRectangular_S16_S16(3MLIB),
mllib_SignalMulRectangular_S16(3MLIB),
mllib_SignalMulWindow_S16(3MLIB),
mllib_SignalMulWindow_S16_S16(3MLIB), attributes(5)

mllib_SignalMulHanning_S16_S16(3MLIB)

NAME mllib_SignalMulHanning_S16_S16, mllib_SignalMulHanning_S16S_S16S – Hanning windowing multiplication

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMulHanning_S16_S16(mllib_s16 *dst, const
mllib_s16 *src, mllib_s32 n);

mllib_status mllib_SignalMulHanning_S16S_S16S(mllib_s16 *dst, const
mllib_s16 *src, mllib_s32 n);
```

DESCRIPTION Each of these functions performs multiplication of the Hanning window.

PARAMETERS Each of the functions takes the following arguments:

dst Destination signal array.

src Source signal array.

n Number of samples in signal and window arrays.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalMulBartlett_S16_S16(3MLIB),
mllib_SignalMulBartlett_S16(3MLIB),
mllib_SignalMulBlackman_S16_S16(3MLIB),
mllib_SignalMulBlackman_S16(3MLIB),
mllib_SignalMulHamming_S16_S16(3MLIB),
mllib_SignalMulHamming_S16(3MLIB), mllib_SignalMulHanning_S16(3MLIB),
mllib_SignalMulKaiser_S16_S16(3MLIB),
mllib_SignalMulKaiser_S16(3MLIB),
mllib_SignalMulRectangular_S16_S16(3MLIB),
mllib_SignalMulRectangular_S16(3MLIB),
mllib_SignalMulWindow_S16_S16(3MLIB), attributes(5)

mllib_SignalMulKaiser_F32(3MLIB)

NAME mllib_SignalMulKaiser_F32, mllib_SignalMulKaiser_F32S – Kaiser windowing multiplication

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMulKaiser_F32(mllib_f32 *srcdst, mllib_f32
    beta, mllib_s32 n);

mllib_status mllib_SignalMulKaiser_F32S(mllib_f32 *srcdst, mllib_f32
    beta, mllib_s32 n);
```

DESCRIPTION Each of these functions performs multiplication of the Kaiser window.

PARAMETERS Each of the functions takes the following arguments:

srcdst Source and destination signal array.

beta Kaiser window parameter.

n Number of samples in signal and window arrays.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalMulBartlett_F32(3MLIB),
mllib_SignalMulBartlett_F32_F32(3MLIB),
mllib_SignalMulBlackman_F32(3MLIB),
mllib_SignalMulBlackman_F32_F32(3MLIB),
mllib_SignalMulHamming_F32(3MLIB),
mllib_SignalMulHamming_F32_F32(3MLIB),
mllib_SignalMulHanning_F32(3MLIB),
mllib_SignalMulHanning_F32_F32(3MLIB),
mllib_SignalMulKaiser_F32_F32(3MLIB),
mllib_SignalMulRectangular_F32_F32(3MLIB),
mllib_SignalMulRectangular_F32(3MLIB),
mllib_SignalMulWindow_F32(3MLIB),
mllib_SignalMulWindow_F32_F32(3MLIB), attributes(5)

mllib_SignalMulKaiser_F32_F32(3MLIB)

NAME mllib_SignalMulKaiser_F32_F32, mllib_SignalMulKaiser_F32S_F32S – Kaiser windowing multiplication

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMulKaiser_F32_F32(mllib_f32 *dst, const
    mllib_f32 *src, mllib_f32 beta, mllib_s32 n);

mllib_status mllib_SignalMulKaiser_F32S_F32S(mllib_f32 *dst, const
    mllib_f32 *src, mllib_f32 beta, mllib_s32 n);
```

DESCRIPTION Each of these functions performs multiplication of the Kaiser window.

PARAMETERS Each of the functions takes the following arguments:

dst Output signal array.
src Input signal array.
beta Kaiser window parameter.
n Number of samples in signal and window arrays.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalMulBartlett_F32(3MLIB),
mllib_SignalMulBartlett_F32_F32(3MLIB),
mllib_SignalMulBlackman_F32(3MLIB),
mllib_SignalMulBlackman_F32_F32(3MLIB),
mllib_SignalMulHamming_F32(3MLIB),
mllib_SignalMulHamming_F32_F32(3MLIB),
mllib_SignalMulHanning_F32(3MLIB),
mllib_SignalMulHanning_F32_F32(3MLIB),
mllib_SignalMulKaiser_F32(3MLIB),
mllib_SignalMulRectangular_F32_F32(3MLIB),
mllib_SignalMulRectangular_F32(3MLIB),
mllib_SignalMulWindow_F32(3MLIB),
mllib_SignalMulWindow_F32_F32(3MLIB), attributes(5)

mllib_SignalMulKaiser_S16(3MLIB)

NAME mllib_SignalMulKaiser_S16, mllib_SignalMulKaiser_S16S – Kaiser windowing multiplication

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMulKaiser_S16(mllib_s16 *srcdst, mllib_f32
    beta, mllib_s32 n);

mllib_status mllib_SignalMulKaiser_S16S(mllib_s16 *srcdst, mllib_f32
    beta, mllib_s32 n);
```

DESCRIPTION Each of these functions performs multiplication of the Kaiser window.

PARAMETERS Each of the functions takes the following arguments:

srcdst Source and destination signal array.

beta Kaiser window parameter.

n Number of samples in signal and window arrays.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalMulBartlett_S16(3MLIB),
mllib_SignalMulBartlett_S16_S16(3MLIB),
mllib_SignalMulBlackman_S16(3MLIB),
mllib_SignalMulBlackman_S16_S16(3MLIB),
mllib_SignalMulHamming_S16(3MLIB),
mllib_SignalMulHamming_S16_S16(3MLIB),
mllib_SignalMulHanning_S16(3MLIB),
mllib_SignalMulHanning_S16_S16(3MLIB),
mllib_SignalMulKaiser_S16_S16(3MLIB),
mllib_SignalMulRectangular_S16(3MLIB),
mllib_SignalMulRectangular_S16_S16(3MLIB),
mllib_SignalMulWindow_S16_S16(3MLIB), attributes(5)

mlib_SignalMulKaiser_S16_S16(3MLIB)

NAME mlib_SignalMulKaiser_S16_S16, mlib_SignalMulKaiser_S16S_S16S – Kaiser windowing multiplication

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_SignalMulKaiser_S16_S16(mlib_s16 *dst, const
    mlib_s16 *src, mlib_f32 beta, mlib_s32 n);

mlib_status mlib_SignalMulKaiser_S16S_S16S(mlib_s16 *dst, const
    mlib_s16 *src, mlib_f32 beta, mlib_s32 n);
```

DESCRIPTION Each of these functions performs multiplication of the Kaiser window.

PARAMETERS Each of the functions takes the following arguments:

dst Output signal array.

src Input signal array.

beta Kaiser window parameter.

n Number of samples in signal and window arrays.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mlib_SignalMulBartlett_S16(3MLIB),
 mlib_SignalMulBartlett_S16_S16(3MLIB),
 mlib_SignalMulBlackman_S16(3MLIB),
 mlib_SignalMulBlackman_S16_S16(3MLIB),
 mlib_SignalMulHamming_S16(3MLIB),
 mlib_SignalMulHamming_S16_S16(3MLIB),
 mlib_SignalMulHanning_S16(3MLIB),
 mlib_SignalMulHanning_S16_S16(3MLIB),
 mlib_SignalMulKaiser_S16(3MLIB),
 mlib_SignalMulRectangular_S16(3MLIB),
 mlib_SignalMulRectangular_S16_S16(3MLIB),
 mlib_SignalMulWindow_S16(3MLIB),
 mlib_SignalMulWindow_S16_S16(3MLIB), attributes(5)

mllib_SignalMulRectangular_F32(3MLIB)

NAME	mllib_SignalMulRectangular_F32, mllib_SignalMulRectangular_F32S – rectangular windowing multiplication						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalMulRectangular_F32(mllib_f32 *srcdst, mllib_s32 m, mllib_s32 n); mllib_status mllib_SignalMulRectangular_F32S(mllib_f32 *srcdst, mllib_s32 m, mllib_s32 n);</pre>						
DESCRIPTION	Each of these functions performs multiplication of the rectangular window.						
PARAMETERS	Each of the functions takes the following arguments: <i>srcdst</i> Input and output signal array. <i>m</i> Rectangular window parameter. <i>n</i> Number of samples in signal and window arrays.						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1" data-bbox="446 1039 1412 1171"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_SignalMulBartlett_F32(3MLIB), mllib_SignalMulBartlett_F32_F32(3MLIB), mllib_SignalMulBlackman_F32(3MLIB), mllib_SignalMulBlackman_F32_F32(3MLIB), mllib_SignalMulHamming_F32(3MLIB), mllib_SignalMulHamming_F32_F32(3MLIB), mllib_SignalMulHanning_F32(3MLIB), mllib_SignalMulHanning_F32_F32(3MLIB), mllib_SignalMulKaiser_F32(3MLIB), mllib_SignalMulKaiser_F32_F32(3MLIB), mllib_SignalMulRectangular_F32_F32(3MLIB), mllib_SignalMulWindow_F32(3MLIB), mllib_SignalMulWindow_F32_F32(3MLIB), attributes(5)						

mllib_SignalMulRectangular_F32_F32(3MLIB)

NAME mllib_SignalMulRectangular_F32_F32, mllib_SignalMulRectangular_F32S_F32S – rectangular windowing multiplication

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMulRectangular_F32_F32(mllib_f32 *dst,
    const mllib_f32 *src, mllib_s32 m, mllib_s32 n);

mllib_status mllib_SignalMulRectangular_F32S_F32S(mllib_f32 *dst,
    const mllib_f32 *src, mllib_s32 m, mllib_s32 n);
```

DESCRIPTION Each of these functions performs multiplication of the Hamming window.

PARAMETERS Each of the functions takes the following arguments:

dst Output signal array.

src Input signal array.

m Rectangular window parameter.

n Number of samples in signal and window arrays.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalMulBartlett_F32(3MLIB),
mllib_SignalMulBartlett_F32_F32(3MLIB),
mllib_SignalMulBlackman_F32(3MLIB),
mllib_SignalMulBlackman_F32_F32(3MLIB),
mllib_SignalMulHamming_F32(3MLIB),
mllib_SignalMulHamming_F32_F32(3MLIB),
mllib_SignalMulHanning_F32(3MLIB),
mllib_SignalMulHanning_F32_F32(3MLIB),
mllib_SignalMulKaiser_F32(3MLIB),
mllib_SignalMulKaiser_F32_F32(3MLIB),
mllib_SignalMulRectangular_F32(3MLIB),
mllib_SignalMulWindow_F32(3MLIB),
mllib_SignalMulWindow_F32_F32(3MLIB), attributes(5)

mllib_SignalMulRectangular_S16(3MLIB)

NAME	mllib_SignalMulRectangular_S16, mllib_SignalMulRectangular_S16S – rectangular windowing multiplication						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalMulRectangular_S16(mllib_s16 *<i>srcdst</i>, mllib_s32 <i>m</i>, mllib_s32 <i>n</i>); mllib_status mllib_SignalMulRectangular_S16S(mllib_s16 *<i>srcdst</i>, mllib_s32 <i>m</i>, mllib_s32 <i>n</i>);</pre>						
DESCRIPTION	Each of these functions performs multiplication of the rectangular window.						
PARAMETERS	Each of the functions takes the following arguments: <p><i>srcdst</i> Input and output signal array.</p> <p><i>m</i> Rectangular window parameter.</p> <p><i>n</i> Number of samples in signal and window arrays.</p>						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<pre>mllib_SignalMulBartlett_S16_S16(3MLIB), mllib_SignalMulBartlett_S16(3MLIB), mllib_SignalMulBlackman_S16_S16(3MLIB), mllib_SignalMulBlackman_S16(3MLIB), mllib_SignalMulHamming_S16_S16(3MLIB), mllib_SignalMulHamming_S16(3MLIB), mllib_SignalMulHanning_S16_S16(3MLIB), mllib_SignalMulHanning_S16(3MLIB), mllib_SignalMulKaiser_S16_S16(3MLIB), mllib_SignalMulKaiser_S16(3MLIB), mllib_SignalMulRectangular_S16_S16(3MLIB), mllib_SignalMulWindow_S16(3MLIB), mllib_SignalMulWindow_S16_S16(3MLIB), attributes(5)</pre>						

mlib_SignalMulRectangular_S16_S16(3MLIB)

NAME | mlib_SignalMulRectangular_S16_S16, mlib_SignalMulRectangular_S16S_S16S – rectangular windowing multiplication

SYNOPSIS |

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_SignalMulRectangular_S16_S16(mlib_s16 *dst,
const mlib_s16 *src, mlib_s32 m, mlib_s32 n);

mlib_status mlib_SignalMulRectangular_S16S_S16S(mlib_s16 *dst,
const mlib_s16 *src, mlib_s32 m, mlib_s32 n);
```

DESCRIPTION | Each of these functions performs multiplication of the Hamming window.

PARAMETERS | Each of the functions takes the following arguments:

dst | Output signal array.

src | Input signal array.

m | Rectangular window parameter.

n | Number of samples in signal and window arrays.

RETURN VALUES | Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mlib_SignalMulBartlett_S16_S16(3MLIB),
mlib_SignalMulBartlett_S16(3MLIB),
mlib_SignalMulBlackman_S16_S16(3MLIB),
mlib_SignalMulBlackman_S16(3MLIB),
mlib_SignalMulHamming_S16_S16(3MLIB),
mlib_SignalMulHamming_S16(3MLIB),
mlib_SignalMulHanning_S16_S16(3MLIB),
mlib_SignalMulHanning_S16(3MLIB),
mlib_SignalMulKaiser_S16_S16(3MLIB),
mlib_SignalMulKaiser_S16(3MLIB),
mlib_SignalMulRectangular_S16(3MLIB),
mlib_SignalMulWindow_S16(3MLIB), mlib_SignalMulWindow_S16(3MLIB),
mlib_SignalMulWindow_S16_S16(3MLIB), attributes(5)

mllib_SignalMul_S16_S16_Sat(3MLIB)

NAME | mllib_SignalMul_S16_S16_Sat, mllib_SignalMul_S16S_S16S_Sat – multiplication

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMul_S16_S16_Sat(mllib_s16 *dst, const
    mllib_s16 *src1, const mllib_s16 *src2, mllib_s32 n);

mllib_status mllib_SignalMul_S16S_S16S_Sat(mllib_s16 *dst, const
    mllib_s16 *src1, const mllib_s16 *src2, mllib_s32 n);
```

DESCRIPTION | Each of these functions performs multiplication.

PARAMETERS | Each of the functions takes the following arguments:

<i>dst</i>	Output signal array.
<i>src1</i>	The first input signal array.
<i>src2</i>	The second input signal array.
<i>n</i>	Number of samples in the input signal arrays.

RETURN VALUES | Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_SignalMul_S16_Sat\(3MLIB\)](#), [attributes\(5\)](#)

NAME mllib_SignalMul_S16_Sat, mllib_SignalMul_S16S_Sat – multiplication

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMul_S16_Sat(mllib_s16 *scr1dst, const
    mllib_s16 *src2, mllib_s32 n);

mllib_status mllib_SignalMul_S16S_Sat(mllib_s16 *scr1dst, const
    mllib_s16 *src2, mllib_s32 n);
```

DESCRIPTION Each of these functions performs multiplication.

PARAMETERS Each of the functions takes the following arguments:

scr1dst The first input and the output signal array.

src2 The second input signal array.

n Number of samples in the input signal arrays.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalMul_S16_S16_Sat(3MLIB), attributes(5)

mllib_SignalMulSAdd_F32(3MLIB)

NAME mllib_SignalMulSAdd_F32, mllib_SignalMulSAdd_F32S – multiplication by a scalar plus addition

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMulSAdd_F32(mllib_f32 *src1dst, const
    mllib_f32 *src2, const mllib_f32 *c, mllib_s32 n);

mllib_status mllib_SignalMulSAdd_F32S(mllib_f32 *src1dst, const
    mllib_f32 *src2, const mllib_f32 *c, mllib_s32 n);
```

DESCRIPTION Each of these functions performs multiplication by a scalar plus addition.

PARAMETERS Each of the functions takes the following arguments:

<i>src1dst</i>	The first input and the output signal array.
<i>src2</i>	The second input signal array.
<i>c</i>	Scaling factor. The scaling factor is in Q15 format. In the stereo version; <i>c</i> [0] contains the scaling factor for channel 0, and <i>c</i> [1] holds the scaling factor for channel 1.
<i>n</i>	Number of samples in the input signal arrays.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalMulSAdd_F32_F32(3MLIB)`, `attributes(5)`

mlib_SignalMulSAdd_F32_F32(3MLIB)

NAME mlib_SignalMulSAdd_F32_F32, mlib_SignalMulSAdd_F32S_F32S – multiplication by a scalar plus addition

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_SignalMulSAdd_F32_F32(mlib_f32 *dst, const
    mlib_f32 *src1, const mlib_f32 *src2, const mlib_f32 *c,
    mlib_s32 n);

mlib_status mlib_SignalMulSAdd_F32S_F32S(mlib_f32 *dst, const
    mlib_f32 *src1, const mlib_f32 *src2, const mlib_f32 *c,
    mlib_s32 n);
```

DESCRIPTION Each of these functions performs multiplication by a scalar.

PARAMETERS Each of the functions takes the following arguments:

dst Output signal array.

src1 The first input signal array.

src2 The second input signal array.

c Scaling factor. The scaling factor is in Q15 format. In the stereo version; *c*[0] contains the scaling factor for channel 0, and *c*[1] holds the scaling factor for channel 1.

n Number of samples in the input signal arrays.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mlib_SignalMulSAdd_F32\(3MLIB\)](#), [attributes\(5\)](#)

mllib_SignalMulSAdd_S16_S16_Sat(3MLIB)

NAME	mllib_SignalMulSAdd_S16_S16_Sat, mllib_SignalMulSAdd_S16S_S16S_Sat – multiplication by a scalar plus addition										
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalMulSAdd_S16_S16_Sat(mllib_s16 *<i>dst</i>, const mllib_s16 *<i>src1</i>, const mllib_s16 *<i>src2</i>, const mllib_s16 *<i>c</i>, mllib_s32 <i>n</i>); mllib_status mllib_SignalMulSAdd_S16S_S16S_Sat(mllib_s16 *<i>dst</i>, const mllib_s16 *<i>src1</i>, const mllib_s16 *<i>src2</i>, const mllib_s16 *<i>c</i>, mllib_s32 <i>n</i>);</pre>										
DESCRIPTION	Each of these functions performs multiplication by a scalar.										
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>dst</i></td><td>Output signal array.</td></tr><tr><td><i>src1</i></td><td>The first input signal array.</td></tr><tr><td><i>src2</i></td><td>The second input signal array.</td></tr><tr><td><i>c</i></td><td>Scaling factor. The scaling factor is in Q15 format. In the stereo version; <i>c</i>[0] contains the scaling factor for channel 0, and <i>c</i>[1] holds the scaling factor for channel 1.</td></tr><tr><td><i>n</i></td><td>Number of samples in the input signal arrays.</td></tr></table>	<i>dst</i>	Output signal array.	<i>src1</i>	The first input signal array.	<i>src2</i>	The second input signal array.	<i>c</i>	Scaling factor. The scaling factor is in Q15 format. In the stereo version; <i>c</i> [0] contains the scaling factor for channel 0, and <i>c</i> [1] holds the scaling factor for channel 1.	<i>n</i>	Number of samples in the input signal arrays.
<i>dst</i>	Output signal array.										
<i>src1</i>	The first input signal array.										
<i>src2</i>	The second input signal array.										
<i>c</i>	Scaling factor. The scaling factor is in Q15 format. In the stereo version; <i>c</i> [0] contains the scaling factor for channel 0, and <i>c</i> [1] holds the scaling factor for channel 1.										
<i>n</i>	Number of samples in the input signal arrays.										
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.										
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe				
ATTRIBUTE TYPE	ATTRIBUTE VALUE										
Interface Stability	Evolving										
MT-Level	MT-Safe										
SEE ALSO	mllib_SignalMulSAdd_S16_Sat(3MLIB) , attributes(5)										

NAME mllib_SignalMulSAdd_S16_Sat, mllib_SignalMulSAdd_S16S_Sat – multiplication by a scalar plus addition

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMulSAdd_S16_Sat(mllib_s16 *src1dst, const
    mllib_s16 *src2, const mllib_s16 *c, mllib_s32 n);

mllib_status mllib_SignalMulSAdd_S16S_Sat(mllib_s16 *src1dst, const
    mllib_s16 *src2, const mllib_s16 *c, mllib_s32 n);
```

DESCRIPTION Each of these functions performs multiplication by a scalar plus addition.

PARAMETERS Each of the functions takes the following arguments:

src1dst The first input and the output signal array.

src2 The second input signal array.

c Scaling factor. The scaling factor is in Q15 format. In the stereo version; *c*[0] contains the scaling factor for channel 0, and *c*[1] holds the scaling factor for channel 1.

n Number of samples in the input signal arrays.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalMulSAdd_S16_S16_Sat(3MLIB)`, `attributes(5)`

mllib_SignalMulS_F32(3MLIB)

NAME	mllib_SignalMulS_F32, mllib_SignalMulS_F32S – multiplication by a scalar						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalMulS_F32(mllib_f32 *<i>srcdst</i>, const mllib_f32 *<i>c</i>, mllib_s32 <i>n</i>); mllib_status mllib_SignalMulS_F32S(mllib_f32 *<i>srcdst</i>, const mllib_f32 *<i>c</i>, mllib_s32 <i>n</i>);</pre>						
DESCRIPTION	Each of these functions performs multiplication by a scalar.						
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>srcdst</i></td><td>Input and output signal array.</td></tr><tr><td><i>c</i></td><td>Scaling factor. The scaling factor is in Q15 format. In the stereo version; <i>c</i>[0] contains the scaling factor for channel 0, and <i>c</i>[1] holds the scaling factor for channel 1.</td></tr><tr><td><i>n</i></td><td>Number of samples in the input signal arrays.</td></tr></table>	<i>srcdst</i>	Input and output signal array.	<i>c</i>	Scaling factor. The scaling factor is in Q15 format. In the stereo version; <i>c</i> [0] contains the scaling factor for channel 0, and <i>c</i> [1] holds the scaling factor for channel 1.	<i>n</i>	Number of samples in the input signal arrays.
<i>srcdst</i>	Input and output signal array.						
<i>c</i>	Scaling factor. The scaling factor is in Q15 format. In the stereo version; <i>c</i> [0] contains the scaling factor for channel 0, and <i>c</i> [1] holds the scaling factor for channel 1.						
<i>n</i>	Number of samples in the input signal arrays.						
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_SignalMulS_F32_F32(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_SignalMulS_F32_F32, mllib_SignalMulS_F32S_F32S – multiplication by a scalar

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMulS_F32_F32(mllib_f32 *dst, const mllib_f32
    *src, const mllib_f32 *c, mllib_s32 n);

mllib_status mllib_SignalMulS_F32S_F32S(mllib_f32 *dst, const
    mllib_f32 *src, const mllib_f32 *c, mllib_s32 n);
```

DESCRIPTION Each of these functions performs multiplication by a scalar.

PARAMETERS Each of the functions takes the following arguments:

dst Output signal array.

src Input signal array.

c Scaling factor. The scaling factor is in Q15 format. In the stereo version; *c*[0] contains the scaling factor for channel 0, and *c*[1] holds the scaling factor for channel 1.

n Number of samples in the input signal arrays.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_SignalMulS_F32\(3MLIB\)](#), `attributes(5)`

mllib_SignalMulShift_S16_S16_Sat(3MLIB)

NAME mllib_SignalMulShift_S16_S16_Sat, mllib_SignalMulShift_S16S_S16S_Sat – multiplication with shifting

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMulShift_S16_S16_Sat(mllib_s16 *dst, const
    mllib_s16 *src1, const mllib_s16 *src2, mllib_s32 shift, mllib_s32 n);

mllib_status mllib_SignalMulShift_S16S_S16S_Sat(mllib_s16 *dst,
    const mllib_s16 *src1, const mllib_s16 *src2, mllib_s32 shift,
    mllib_s32 n);
```

DESCRIPTION Each of these functions performs multiplication with shifting.

PARAMETERS Each of the functions takes the following arguments:

<i>dst</i>	Output signal array.
<i>src1</i>	The first input signal array.
<i>src2</i>	The second input signal array.
<i>shift</i>	Left shifting factor.
<i>n</i>	Number of samples in the input signal arrays.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalMulShift_S16_Sat(3MLIB), attributes(5)

mllib_SignalMulShift_S16_Sat(3MLIB)

NAME mllib_SignalMulShift_S16_Sat, mllib_SignalMulShift_S16S_Sat – multiplication with shifting

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
mllib_status mllib_SignalMulShift_S16_Sat(mllib_s16 *src1dst, const  
mllib_s16 *src2, mllib_s32 shift, mllib_s32 n);  
  
mllib_status mllib_SignalMulShift_S16S_Sat(mllib_s16 *src1dst, const  
mllib_s16 *src2, mllib_s32 shift, mllib_s32 n);
```

DESCRIPTION Each of these functions performs multiplication with shifting.

PARAMETERS Each of the functions takes the following arguments:

src1dst The first input and the output signal array.
src2 The second input signal array.
shift Left shifting factor.
n Number of samples in the input signal arrays.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_SignalMulShift_S16_S16_Sat\(3MLIB\)](#), [attributes\(5\)](#)

mllib_SignalMulS_S16_S16_Sat(3MLIB)

NAME mllib_SignalMulS_S16_S16_Sat, mllib_SignalMulS_S16S_S16S_Sat – multiplication by a scalar

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMulS_S16_S16_Sat(mllib_s16 *dst, const
    mllib_s16 *src, const mllib_s16 *c, mllib_s32 n);

mllib_status mllib_SignalMulS_S16S_S16S_Sat(mllib_s16 *dst, const
    mllib_s16 *src, const mllib_s16 *c, mllib_s32 n);
```

DESCRIPTION Each of these functions performs multiplication by a scalar.

PARAMETERS Each of the functions takes the following arguments:

<i>dst</i>	Output signal array.
<i>src</i>	Input signal array.
<i>c</i>	Scaling factor. The scaling factor is in Q15 format. In the stereo version; <i>c</i> [0] contains the scaling factor for channel 0, and <i>c</i> [1] holds the scaling factor for channel 1.
<i>n</i>	Number of samples in the input signal arrays.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_SignalMulS_S16_Sat\(3MLIB\)](#), [attributes\(5\)](#)

mlib_SignalMulS_S16_Sat(3MLIB)

NAME | mlib_SignalMulS_S16_Sat, mlib_SignalMulS_S16S_Sat – multiplication by a scalar

SYNOPSIS |

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_SignalMulS_S16_Sat(mlib_s16 *srcdst, const
    mlib_s16 *c, mlib_s32 n);

mlib_status mlib_SignalMulS_S16S_Sat(mlib_s16 *srcdst, const
    mlib_s16 *c, mlib_s32 n);
```

DESCRIPTION | Each of these functions performs multiplication by a scalar.

PARAMETERS | Each of the functions takes the following arguments:

srcdst | Input and output signal array.

c | Scaling factor. The scaling factor is in Q15 format. In the stereo version; c[0] contains the scaling factor for channel 0, and c[1] holds the scaling factor for channel 1.

n | Number of samples in the input signal arrays.

RETURN VALUES | Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mlib_SignalMulS_S16_S16_Sat\(3MLIB\)](#), [attributes\(5\)](#)

mllib_SignalMulSShiftAdd_S16_S16_Sat(3MLIB)

NAME	mllib_SignalMulSShiftAdd_S16_S16_Sat, mllib_SignalMulSShiftAdd_S16S_S16S_Sat – multiplication by a scalar plus addition												
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalMulSShiftAdd_S16_S16_Sat(mllib_s16 *dst, const mllib_s16 *src1, const mllib_s16 *src2, const mllib_s16 *c, mllib_s32 shift, mllib_s32 n); mllib_status mllib_SignalMulSShiftAdd_S16S_S16S_Sat(mllib_s16 *dst, const mllib_s16 *src1, const mllib_s16 *src2, const mllib_s16 *c, mllib_s32 shift, mllib_s32 n);</pre>												
DESCRIPTION	Each of these functions performs multiplication by a scalar with shifting plus addition.												
PARAMETERS	Each of the functions takes the following arguments: <table><tr><td><i>dst</i></td><td>Output signal array.</td></tr><tr><td><i>src1</i></td><td>The first input signal array.</td></tr><tr><td><i>src2</i></td><td>The second input signal array.</td></tr><tr><td><i>c</i></td><td>Scaling factor. The scaling factor is in Q15 format. In the stereo version; c[0] contains the scaling factor for channel 0, and c[1] holds the scaling factor for channel 1.</td></tr><tr><td><i>shift</i></td><td>Left shifting factor.</td></tr><tr><td><i>n</i></td><td>Number of samples in the input signal arrays.</td></tr></table>	<i>dst</i>	Output signal array.	<i>src1</i>	The first input signal array.	<i>src2</i>	The second input signal array.	<i>c</i>	Scaling factor. The scaling factor is in Q15 format. In the stereo version; c[0] contains the scaling factor for channel 0, and c[1] holds the scaling factor for channel 1.	<i>shift</i>	Left shifting factor.	<i>n</i>	Number of samples in the input signal arrays.
<i>dst</i>	Output signal array.												
<i>src1</i>	The first input signal array.												
<i>src2</i>	The second input signal array.												
<i>c</i>	Scaling factor. The scaling factor is in Q15 format. In the stereo version; c[0] contains the scaling factor for channel 0, and c[1] holds the scaling factor for channel 1.												
<i>shift</i>	Left shifting factor.												
<i>n</i>	Number of samples in the input signal arrays.												
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.												
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe						
ATTRIBUTE TYPE	ATTRIBUTE VALUE												
Interface Stability	Evolving												
MT-Level	MT-Safe												
SEE ALSO	mllib_SignalMulSShiftAdd_S16_Sat(3MLIB), attributes(5)												

mlib_SignalMulSShiftAdd_S16_Sat(3MLIB)

NAME mlib_SignalMulSShiftAdd_S16_Sat, mlib_SignalMulSShiftAdd_S16S_Sat – multiplication by a scalar plus addition

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_SignalMulSShiftAdd_S16_Sat(mlib_s16 *src1dst,
      const mlib_s16 *src2, const mlib_s16 *c, mlib_s32 shift, mlib_s32
      n);

mlib_status mlib_SignalMulSShiftAdd_S16S_Sat(mlib_s16 *src1dst,
      const mlib_s16 *src2, const mlib_s16 *c, mlib_s32 shift, mlib_s32
      n);
```

DESCRIPTION Each of these functions performs multiplication by a scalar with shifting plus addition.

PARAMETERS Each of the functions takes the following arguments:

- src1dst* The first input and the output signal array.
- src2* The second input signal array.
- c* Scaling factor. The scaling factor is in Q15 format. In the stereo version; *c*[0] contains the scaling factor for channel 0, and *c*[1] holds the scaling factor for channel 1.
- n* Number of samples in the input signal arrays.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mlib_SignalMulSShiftAdd_S16_S16_Sat(3MLIB), attributes(5)

mllib_SignalMulSShift_S16_S16_Sat(3MLIB)

NAME	mllib_SignalMulSShift_S16_S16_Sat, mllib_SignalMulSShift_S16S_S16S_Sat – multiplication by a scalar with shifting						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalMulSShift_S16_S16_Sat(mllib_s16 *<i>dst</i>, const mllib_s16 *<i>src</i>, const mllib_s16 *<i>c</i>, mllib_s32 <i>shift</i>, mllib_s32 <i>n</i>); mllib_status mllib_SignalMulSShift_S16S_S16S_Sat(mllib_s16 *<i>dst</i>, const mllib_s16 *<i>src</i>, const mllib_s16 *<i>c</i>, mllib_s32 <i>shift</i>, mllib_s32 <i>n</i>);</pre>						
DESCRIPTION	Each of these functions performs multiplication by a scalar with shifting.						
PARAMETERS	Each of the functions takes the following arguments: <i>dst</i> Destination signal array. <i>src</i> Source signal array. <i>c</i> Scaling factor. The scaling factor is in Q15 format. In the stereo version; <i>c</i> [0] contains the scaling factor for channel 0, and <i>c</i> [1] holds the scaling factor for channel 1. <i>shift</i> Left shifting factor. <i>n</i> Number of samples in the input signal arrays.						
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_SignalMulSShift_S16_Sat(3MLIB) , attributes(5)						

NAME mllib_SignalMulSShift_S16_Sat, mllib_SignalMulSShift_S16S_Sat – multiplication by a scalar with shifting

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMulSShift_S16_Sat(mllib_s16 *srcdst, const
      mllib_s16 *c, mllib_s32 shift, mllib_s32 n);

mllib_status mllib_SignalMulSShift_S16S_Sat(mllib_s16 *srcdst, const
      mllib_s16 *c, mllib_s32 shift, mllib_s32 n);
```

DESCRIPTION Each of these functions performs multiplication by a scalar with shifting.

PARAMETERS Each of the functions takes the following arguments:

srcdst Source and destination signal array.

c Scaling factor. The scaling factor is in Q15 format. In the stereo version; *c*[0] contains the scaling factor for channel 0, and *c*[1] holds the scaling factor for channel 1.

shift Left shifting factor.

n Number of samples in the input signal arrays.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalMulSShift_S16_S16_Sat(3MLIB)`, `attributes(5)`

mllib_SignalMulWindow_F32(3MLIB)

NAME | mllib_SignalMulWindow_F32 – windowing

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
mllib_status mllib_SignalMulWindow_F32(mllib_f32 *srcdst, const  
mllib_f32 *window, mllib_s32 n);
```

DESCRIPTION | The mllib_SignalMulWindow_F32() function performs a windowing operation.

PARAMETERS | The function takes the following arguments:

<i>srcdst</i>	Input and output signal array.
<i>window</i>	Window coefficient array. The window coefficients are in Q15 format.
<i>n</i>	Number of samples in signal and window arrays.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | attributes(5)

mllib_SignalMulWindow_F32_F32(3MLIB)

NAME | mllib_SignalMulWindow_F32_F32 – windowing

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMulWindow_F32_F32(mllib_f32 *dst, const
mllib_f32 *src, const mllib_f32 *window, mllib_s32 n);
```

DESCRIPTION | The `mllib_SignalMulWindow_F32_F32()` function performs a windowing operation.

PARAMETERS | The function takes the following arguments:

dst | Output signal array.

src | Input signal array.

window | Window coefficient array. The window coefficients are in Q15 format.

n | Number of samples in signal and window arrays.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `attributes(5)`

mllib_SignalMulWindow_F32S(3MLIB)

NAME | mllib_SignalMulWindow_F32S – windowing

SYNOPSIS |

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMulWindow_F32S(mllib_f32 *srcdst, const
mllib_f32 *window, mllib_s32 n);
```

DESCRIPTION | The mllib_SignalMulWindow_F32S() function performs a windowing operation.

PARAMETERS | The function takes the following arguments:

srcdst | Input and output signal array are in stereo format where srcdst[0] contains the values for channel 0, and srcdst[1] holds the values for channel 1.

window | Window coefficient array. The window coefficients are in Q15 format.

n | Number of samples in signal and window arrays.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | attributes(5)

mllib_SignalMulWindow_F32S_F32S(3MLIB)

NAME mllib_SignalMulWindow_F32S_F32S – windowing

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMulWindow_F32S_F32S(mllib_f32 *dst, const
mllib_f32 *src, const mllib_f32 *window, mllib_s32 n);
```

DESCRIPTION The mllib_SignalMulWindow_F32S_F32S() function performs a windowing operation.

PARAMETERS The function takes the following arguments:

dst Output signal array is in stereo format where dst[0] contains the values for channel 0, and dst[1] holds the valuesfor channel 1.

src Input signal array is in stereo format where src[0] contains the values for channel 0, and src[1] holds the valuesfor channel 1.

window Window coefficient array. The window coefficients are in Q15 format.

n Number of samples in signal and window arrays.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO attributes(5)

mllib_SignalMulWindow_S16(3MLIB)

NAME mllib_SignalMulWindow_S16, mllib_SignalMulWindow_S16S – windowing

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMulWindow_S16(mllib_s16 *srcdst, const
    mllib_s16 *window, mllib_s32 n);

mllib_status mllib_SignalMulWindow_S16S(mllib_s16 *srcdst, const
    mllib_s16 *window, mllib_s32 n);
```

DESCRIPTION Each of these functions performs a windowing operation.

PARAMETERS Each of the functions takes the following arguments:

srcdst Input and output signal array.

window Window coefficient array. The window coefficients are in Q15 format.

n Number of samples in signal and window arrays.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalMulBartlett_S16(3MLIB),
mllib_SignalMulBartlett_S16_S16(3MLIB),
mllib_SignalMulBlackman_S16(3MLIB),
mllib_SignalMulBlackman_S16_S16(3MLIB),
mllib_SignalMulHamming_S16(3MLIB),
mllib_SignalMulHamming_S16_S16(3MLIB),
mllib_SignalMulHanning_S16(3MLIB),
mllib_SignalMulHanning_S16_S16(3MLIB),
mllib_SignalMulKaiser_S16(3MLIB),
mllib_SignalMulKaiser_S16_S16(3MLIB),
mllib_SignalMulRectangular_S16(3MLIB),
mllib_SignalMulRectangular_S16_S16(3MLIB),
mllib_SignalMulWindow_S16_S16(3MLIB), attributes(5)

mllib_SignalMulWindow_S16_S16(3MLIB)

NAME | mllib_SignalMulWindow_S16_S16, mllib_SignalMulWindow_S16S_S16S – windowing

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalMulWindow_S16_S16(mllib_s16 *dst, const
mllib_s16 *src, const mllib_s16 *window, mllib_s32 n);

mllib_status mllib_SignalMulWindow_S16S_S16S(mllib_s16 *dst, const
mllib_s16 *src, const mllib_s16 *window, mllib_s32 n);
```

DESCRIPTION | Each of these functions performs a windowing operation.

PARAMETERS | Each of the functions takes the following arguments:

dst | Output signal array.

src | Input signal array.

window | Window coefficient array. The window coefficients are in Q15 format.

n | Number of samples in signal and window arrays.

RETURN VALUES | Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_SignalMulBartlett_S16(3MLIB),
mllib_SignalMulBartlett_S16_S16(3MLIB),
mllib_SignalMulBlackman_S16(3MLIB),
mllib_SignalMulBlackman_S16_S16(3MLIB),
mllib_SignalMulHamming_S16(3MLIB),
mllib_SignalMulHamming_S16_S16(3MLIB),
mllib_SignalMulHanning_S16(3MLIB),
mllib_SignalMulHanning_S16_S16(3MLIB),
mllib_SignalMulKaiser_S16(3MLIB),
mllib_SignalMulKaiser_S16_S16(3MLIB),
mllib_SignalMulRectangular_S16(3MLIB),
mllib_SignalMulRectangular_S16_S16(3MLIB),
mllib_SignalMulWindow_S16(3MLIB), attributes(5)

mllib_SignalQuant2_S16_F32(3MLIB)

NAME	mllib_SignalQuant2_S16_F32 – float to 16-bit quantization						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalQuant2_S16_F32(mllib_s16 *<i>dst</i>, const mllib_f32 *<i>src</i>, const mllib_f32 <i>thresh</i>, mllib_s32 <i>length</i>, mllib_s16 <i>offset</i>, mllib_s32 <i>n</i>);</pre>						
DESCRIPTION	<p>The mllib_SignalQuant2_S16_F32() function quantizes a signal array by using the following equation:</p> $\begin{aligned} X &= x(n) && n = 0, 1, \dots \\ Z &= z(n) && n = 0, 1, \dots \\ &= \text{offset} && \text{for } x(n) < t(0) \\ &= \text{offset} + k && \text{for } t(k) \leq x(n) < t(k+1) \\ &= \text{offset} + \text{length} - 1 && \text{for } x(n) \geq t(\text{length} - 1) \end{aligned}$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Output signal array</p> <p><i>src</i> Input signal array .</p> <p><i>thresh</i> Array of thresholds.</p> <p><i>length</i> Length of the array of thresholds.</p> <p><i>offset</i> Offset for thresholds.</p> <p><i>n</i> Number of samples in the input signal array.</p>						
RETURN VALUES	The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	attributes(5)						

NAME mllib_SignalQuant2_S16S_F32S – float to 16-bit quantization

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalQuant2_S16S_F32S(mllib_s16 *dst, const
      mllib_f32 *src, const mllib_f32 thresh, mllib_s32 length, mllib_s16
      offset, mllib_s32 n);
```

DESCRIPTION The `mllib_SignalQuant2_S16S_F32S()` function quantizes a signal array by using the following equation:

$$\begin{aligned}
 X &= x(n) & n &= 0, 1, \dots \\
 Z &= z(n) & n &= 0, 1, \dots \\
 &= \text{offset} & \text{for } x(n) < t(0) \\
 &= \text{offset} + k & \text{for } t(k) \leq x(n) < t(k+1) \\
 &= \text{offset} + \text{length} - 1 & \text{for } x(n) \geq t(\text{length} - 1)
 \end{aligned}$$

PARAMETERS The function takes the following arguments:

dst Output signal array in two-channel interleaved stereo format.

src Input signal array in two-channel interleaved stereo format.

thresh Array of thresholds.

length Length of the array of thresholds.

offset Offset for thresholds.

n Number of samples in the input signal array.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `attributes(5)`

mllib_SignalQuant_S16_F32(3MLIB)

NAME	mllib_SignalQuant_S16_F32 – float to 16-bit quantization						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_SignalQuant_S16_F32(mllib_s16 *dst, const mllib_f32 *src, const mllib_f32 *thresh, mllib_s32 n);</pre>						
DESCRIPTION	<p>The <code>mllib_SignalQuant_S16_F32()</code> function quantizes a signal array by using the following equation:</p> $\begin{aligned} X &= x(n) & n &= 0, 1, \dots \\ Z &= z(n) & n &= 0, 1, \dots \\ &= -32768 & \text{for } x(n) < t(-32768) \\ &= k & \text{for } t(k) \leq x(n) < t(k+1) \\ &= +32767 & \text{for } x(n) \geq t(+32767) \end{aligned}$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Output signal array</p> <p><i>src</i> Input signal array .</p> <p><i>thresh</i> Array of 65536 thresholds.</p> <p><i>n</i> Number of samples in the input signal array.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>attributes(5)</code>						

NAME mllib_SignalQuant_S16S_F32S – float to 16-bit quantization

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalQuant_S16S_F32S(mllib_s16 *dst, const
mllib_f32 *src, const mllib_f32 *thresh, mllib_s32 n);
```

DESCRIPTION The mllib_SignalQuant_S16S_F32S() function quantizes a signal array by using the following equation:

$$\begin{aligned}
 X &= x(n) & n &= 0, 1, \dots \\
 Z &= z(n) & n &= 0, 1, \dots \\
 &= -32768 & \text{for } x(n) < t(-32768) \\
 &= k & \text{for } t(k) \leq x(n) < t(k+1) \\
 &= +32767 & \text{for } x(n) \geq t(+32767)
 \end{aligned}$$

PARAMETERS The function takes the following arguments:

- dst* Output signal array in two-channel interleaved stereo format.
- src* Input signal array in two-channel interleaved stereo format.
- thresh* Array of 65536 thresholds.
- n* Number of samples in the input signal array.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO attributes(5)

mllib_SignalQuant_U8_F32(3MLIB)

NAME	mllib_SignalQuant_U8_F32 – float to 8-bit quantization						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalQuant_U8_F32(mllib_u8 *<i>dst</i>, const mllib_f32 *<i>src</i>, const mllib_f32 *<i>thresh</i>, mllib_s32 <i>n</i>);</pre>						
DESCRIPTION	<p>The mllib_SignalQuant_U8_F32() function quantizes a signal array by using the following equation:</p> $\begin{aligned} X &= x(n) & n &= 0, 1, \dots \\ Z &= z(n) & n &= 0, 1, \dots \\ &= 0 & \text{for } x(n) < t(0) \\ &= k & \text{for } t(k) \leq x(n) < t(k+1) \\ &= 255 & \text{for } x(n) \geq t(255) \end{aligned}$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Output signal array</p> <p><i>src</i> Input signal array .</p> <p><i>thresh</i> Array of 256 thresholds.</p> <p><i>n</i> Number of samples in the input signal array.</p>						
RETURN VALUES	The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	attributes(5)						

NAME | mllib_SignalQuant_U8_S16, mllib_SignalQuant_U8S_S16S – 16-bit to 8-bit quantization

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalQuant_U8_S16(mllib_u8 *dst, const mllib_s16
    *src, const mllib_s16 *thresh, mllib_s32 n);

mllib_status mllib_SignalQuant_U8S_S16S(mllib_u8 *dst, const
    mllib_s16 *src, const mllib_s16 *thresh, mllib_s32 n);
```

DESCRIPTION | Each of these functions quantizes a signal array by using the following equation:

$X = x(n) \quad n = 0, 1, \dots$
 $Z = z(n) \quad n = 0, 1, \dots$
 $= 0 \quad \text{for } x(n) < t(0)$
 $= k \quad \text{for } t(k) \leq x(n) < t(k+1)$
 $= 255 \quad \text{for } x(n) \geq t(255)$

PARAMETERS | Each of the functions takes the following arguments:

dst Output signal array.

src Input signal array.

thresh Array of 256 thresholds.

n Number of samples in the input signal array.

RETURN VALUES | Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | attributes(5)

mllib_SignalQuant_U8S_F32S(3MLIB)

NAME	mllib_SignalQuant_U8S_F32S – float to 8-bit quantization						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalQuant_U8S_F32S(mllib_u8 *<i>dst</i>, const mllib_f32 *<i>src</i>, const mllib_f32 *<i>thresh</i>, mllib_s32 <i>n</i>);</pre>						
DESCRIPTION	<p>The mllib_SignalQuant_U8S_F32S() function quantizes a signal array by using the following equation:</p> $X = x(n) \quad n = 0, 1, \dots$ $Z = z(n) \quad n = 0, 1, \dots$ $= 0 \quad \text{for } x(n) < t(0)$ $= k \quad \text{for } t(k) \leq x(n) < t(k+1)$ $= 255 \quad \text{for } x(n) \geq t(255)$						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>dst</i> Output signal array in two-channel interleaved stereo format.</p> <p><i>src</i> Input signal array in two-channel interleaved stereo format.</p> <p><i>thresh</i> Array of 256 thresholds.</p> <p><i>n</i> Number of samples in the input signal array.</p>						
RETURN VALUES	The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	attributes(5)						

mllib_SignalReSampleFIR_F32_F32(3MLIB)

NAME mllib_SignalReSampleFIR_F32_F32 – resampling with filtering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalReSampleFIR_F32_F32(mllib_f32 *dst, const
mllib_f32 *src, void *state, mllib_s32 n);
```

DESCRIPTION The mllib_SignalReSampleFIR_F32_F32() function performs rational sample rate conversion with FIR filtering between the upsampling and downsampling.

PARAMETERS The function takes the following arguments:

dst Output signal array.

src Input signal array.

state Internal state structure.

n Number of samples in the input signal array.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO attributes(5)

mllib_SignalReSampleFIR_F32S_F32S(3MLIB)

NAME | mllib_SignalReSampleFIR_F32S_F32S – resampling with filtering

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalReSampleFIR_F32S_F32S(mllib_f32 *dst, const
mllib_f32 *src, void *state, mllib_s32 n);
```

DESCRIPTION | The mllib_SignalReSampleFIR_F32S_F32S() function performs rational sample rate conversion with FIR filtering between the upsampling and downsampling.

PARAMETERS | The function takes the following arguments:

dst | Output signal array in two-channel interleaved stereo format.

src | Input signal array in two-channel interleaved stereo format.

state | Internal state structure.

n | Number of samples in the input signal array.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | attributes(5)

mllib_SignalReSampleFIRFree_F32_F32(3MLIB)

NAME mllib_SignalReSampleFIRFree_F32_F32 – resampling with filtering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
void mllib_SignalReSampleFIRFree_F32_F32(void *state);
```

DESCRIPTION The mllib_SignalReSampleFIRFree_F32_F32() function releases the memory allocated for the internal state structure for rational sample rate conversion with FIR filtering between upsampling and downsampling.

PARAMETERS The function takes the following arguments:
state Internal state structure.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO attributes(5)

mllib_SignalReSampleFIRFree_F32S_F32S(3MLIB)

NAME | mllib_SignalReSampleFIRFree_F32S_F32S – resampling with filtering

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
void mllib_SignalReSampleFIRFree_F32S_F32S(void *state);
```

DESCRIPTION | The `mllib_SignalReSampleFIRFree_F32S_F32S()` function releases the memory allocated for the internal state structure for rational sample rate conversion with FIR filtering between upsampling and downsampling.

PARAMETERS | The function takes the following arguments:
state Internal state structure.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `attributes(5)`

mllib_SignalReSampleFIRFree_S16_S16(3MLIB)

NAME mllib_SignalReSampleFIRFree_S16_S16, mllib_SignalReSampleFIRFree_S16S_S16S – resampling with filtering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
void mllib_SignalReSampleFIRFree_S16_S16(void *state);  
void mllib_SignalReSampleFIRFree_S16S_S16S(void *state);
```

DESCRIPTION Each of these functions releases the memory allocated for the internal state structure for rational sample rate conversion with FIR filtering between upsampling and downsampling.

PARAMETERS Each of the functions takes the following arguments:
state Internal state structure.

RETURN VALUES None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalReSampleFIR_S16_S16_Sat(3MLIB),
mllib_SignalReSampleFIRInit_S16_S16(3MLIB), attributes(5)

mllib_SignalReSampleFIRInit_S16_S16(3MLIB)

NAME	mllib_SignalReSampleFIRInit_S16_S16, mllib_SignalReSampleFIRInit_S16S_S16S, mllib_SignalReSampleFIRInit_F32_F32, mllib_SignalReSampleFIRInit_F32S_F32S – initialization for resampling with filtering														
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalReSampleFIRInit_S16_S16(void **state, const mllib_f32 *flt, mllib_s32 tap, mllib_s32 ufactor, mllib_s32 uphase, mllib_s32 dfactor, mllib_s32 dphase); mllib_status mllib_SignalReSampleFIRInit_S16S_S16S(void **state, const mllib_f32 *flt, mllib_s32 tap, mllib_s32 ufactor, mllib_s32 uphase, mllib_s32 dfactor, mllib_s32 dphase); mllib_status mllib_SignalReSampleFIRInit_F32_F32(void **state, const mllib_f32 *flt, mllib_s32 tap, mllib_s32 ufactor, mllib_s32 uphase, mllib_s32 dfactor, mllib_s32 dphase); mllib_status mllib_SignalReSampleFIRInit_F32S_F32S(void **state, const mllib_f32 *flt, mllib_s32 tap, mllib_s32 ufactor, mllib_s32 uphase, mllib_s32 dfactor, mllib_s32 dphase);</pre>														
DESCRIPTION	Each of these functions allocates memory for the internal state structure and converts the parameters into an internal representation for rational sample rate conversion with FIR filtering between upsampling and downsampling.														
PARAMETERS	Each of the functions takes the following arguments: <table border="0" style="margin-left: 20px;"> <tr> <td style="padding-right: 20px;"><i>state</i></td> <td>Internal state structure.</td> </tr> <tr> <td><i>flt</i></td> <td>Filter coefficient array, two-channel interleaved in the cases of stereo.</td> </tr> <tr> <td><i>tap</i></td> <td>Taps of the filter.</td> </tr> <tr> <td><i>ufactor</i></td> <td>Factor by which to upsample.</td> </tr> <tr> <td><i>uphase</i></td> <td>Phase in upsampling. $0 \leq \text{uphase} < \text{ufactor}$.</td> </tr> <tr> <td><i>dfactor</i></td> <td>Factor by which to downsample.</td> </tr> <tr> <td><i>dphase</i></td> <td>Phase in downsampling. $0 \leq \text{dphase} < \text{dfactor}$.</td> </tr> </table>	<i>state</i>	Internal state structure.	<i>flt</i>	Filter coefficient array, two-channel interleaved in the cases of stereo.	<i>tap</i>	Taps of the filter.	<i>ufactor</i>	Factor by which to upsample.	<i>uphase</i>	Phase in upsampling. $0 \leq \text{uphase} < \text{ufactor}$.	<i>dfactor</i>	Factor by which to downsample.	<i>dphase</i>	Phase in downsampling. $0 \leq \text{dphase} < \text{dfactor}$.
<i>state</i>	Internal state structure.														
<i>flt</i>	Filter coefficient array, two-channel interleaved in the cases of stereo.														
<i>tap</i>	Taps of the filter.														
<i>ufactor</i>	Factor by which to upsample.														
<i>uphase</i>	Phase in upsampling. $0 \leq \text{uphase} < \text{ufactor}$.														
<i>dfactor</i>	Factor by which to downsample.														
<i>dphase</i>	Phase in downsampling. $0 \leq \text{dphase} < \text{dfactor}$.														
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.														
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:														
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td style="text-align: center;">Interface Stability</td> <td style="text-align: center;">Evolving</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving										
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														

mllib_SignalReSampleFIRInit_S16_S16(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	MT-Safe

SEE ALSO

mllib_SignalReSampleFIR_S16_S16_Sat(3MLIB),
mllib_SignalReSampleFIR_F32_F32(3MLIB),
mllib_SignalReSampleFIRFree_S16_S16(3MLIB),
mllib_SignalReSampleFIRFree_F32_F32(3MLIB), attributes(5)

mllib_SignalReSampleFIR_S16_S16_Sat(3MLIB)

NAME | mllib_SignalReSampleFIR_S16_S16_Sat, mllib_SignalReSampleFIR_S16S_S16S_Sat – resampling with filtering

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalReSampleFIR_S16_S16_Sat(mllib_s16 *dst,
const mllib_s16 *src, void *state, mllib_s32 n);

mllib_status mllib_SignalReSampleFIR_S16S_S16S_Sat(mllib_s16 *dst,
const mllib_s16 *src, void *state, mllib_s32 n);
```

DESCRIPTION | Each of these functions performs rational sample rate conversion with FIR filtering between the upsampling and downsampling.

PARAMETERS | Each of the functions takes the following arguments:

dst | Output signal array.

src | Input signal array.

state | Internal state structure.

n | Number of samples in the input signal array.

RETURN VALUES | Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `attributes(5)`

NAME mllib_SignalSineWave_F32 – sine wave generation

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalSineWave_F32(mllib_f32 *sine, void *state,
mllib_s32 n);
```

DESCRIPTION The mllib_SignalSineWave_F32() function generates one packet of sine wave and updates the internal state.

PARAMETERS The function takes the following arguments:

sine Generated sine wave array.

state Internal state structure.

n Length of the generated sine wave array in number of samples.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO attributes(5)

mllib_SignalSineWaveFree_F32(3MLIB)

NAME | mllib_SignalSineWaveFree_F32 – sine wave generation

SYNOPSIS | cc [*flag...*] *file...* -lmllib [*library...*]
#include <mllib.h>

```
void mllib_SignalSineWaveFree_F32(void *state);
```

DESCRIPTION | The mllib_SignalSineWaveFree_F32() function releases the memory allocated for the internal state's structure.

PARAMETERS | The function takes the following arguments:
state Internal state structure.

RETURN VALUES | None.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_SignalSineWave_S16(3MLIB), mllib_SignalSineWaveInit_S16(3MLIB), attributes(5)

NAME | mllib_SignalSineWaveFree_S16 – sine wave generation

SYNOPSIS | `cc [flag...] file... -lmllib [library...]`
`#include <mllib.h>`
`void mllib_SignalSineWaveFree_S16(void *state) ;`

DESCRIPTION | The mllib_SignalSineWaveFree_S16() function releases the memory allocated for the internal state's structure.

PARAMETERS | The function takes the following arguments:
state Internal state structure.

RETURN VALUES | None.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | attributes(5)

mllib_SignalSineWaveInit_F32(3MLIB)

NAME | mllib_SignalSineWaveInit_F32 – sine wave generation

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalSineWaveInit_F32(void **state, mllib_f32 mag,
      mllib_f32 freq, mllib_f32 phase);
```

DESCRIPTION | The `mllib_SignalSineWaveInit_F32()` function allocates memory for an internal state structure and converts the parameters of the wave to an internal representation.

PARAMETERS | The function takes the following arguments:

<i>state</i>	Internal state structure.
<i>mag</i>	Magnitude of sine wave to be generated, in Q15 format.
<i>freq</i>	Angular frequency of the sine wave to be generated, measured in radians per sample.
<i>phase</i>	Start phase of the sine wave to be generated, measured in radians.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `attributes(5)`

NAME mllib_SignalSineWaveInit_S16 – sine wave generation

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalSineWaveInit_S16(void **state, mllib_s16 mag,
      mllib_f32 freq, mllib_f32 phase);
```

DESCRIPTION The `mllib_SignalSineWaveInit_S16()` function allocates memory for an internal state structure and converts the parameters of the wave to an internal representation.

PARAMETERS The function takes the following arguments:

state Internal state structure.

mag Magnitude of sine wave to be generated, in Q15 format.

freq Angular frequency of the sine wave to be generated, measured in radians per sample.

phase Start phase of the sine wave to be generated, measured in radians.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalSineWave_S16(3MLIB)`, `mllib_SignalSineWaveFree_S16(3MLIB)`, `attributes(5)`

mllib_SignalSineWave_S16(3MLIB)

NAME | mllib_SignalSineWave_S16 – sine wave generation

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalSineWave_S16(mllib_s16 *sine, void *state,
mllib_s32 n);
```

DESCRIPTION | The `mllib_SignalSineWave_S16()` function generates one packet of sine wave and updates the internal state.

PARAMETERS | The function takes the following arguments:

sine | Generated sine wave array.

state | Internal state structure.

n | Length of the generated sine wave array in number of samples.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_SignalSineWaveFree_S16\(3MLIB\)](#),
[mllib_SignalSineWaveInit_S16\(3MLIB\)](#), `attributes(5)`

NAME mlib_SignalSplit_F32_F32S – split

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_SignalSplit_F32_F32S(mlib_f32 *ch0, mlib_f32
    *ch1, const mlib_f32 *src, mlib_s32 n);
```

DESCRIPTION The following function splits a stereo signal array into two signal arrays.

PARAMETERS The function takes the following arguments:

ch0 Destination signal array of Channel 0.

ch1 Destination signal array of Channel 1.

src Source stereo signal array. *src*[2**i*] contains Channel 0, and *src*[2**i*+1] contains Channel 1.

n Number of samples in the source signal array.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO attributes(5)

mllib_SignalSplit_S16_S16S(3MLIB)

NAME | mllib_SignalSplit_S16_S16S – split

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalSplit_S16_S16S(mllib_s16 *ch0, mllib_s16
    *ch1, const mllib_s16 *src, mllib_s32 n);
```

DESCRIPTION | The following function splits a stereo signal array into two signal arrays.

PARAMETERS | The function takes the following arguments:

<i>ch0</i>	Destination signal array of Channel 0.
<i>ch1</i>	Destination signal array of Channel 1.
<i>src</i>	Source stereo signal array. <i>src</i> [2* <i>i</i>] contains Channel 0, and <i>src</i> [2* <i>i</i> +1] contains Channel 1.
<i>n</i>	Number of samples in the source signal array.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_SignalMerge_S16S_S16(3MLIB)`, `attributes(5)`

NAME mlib_SignaluLaw2ALaw – ITU G.711 m-law and A-law compression and decompression

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_SignaluLaw2ALaw(mlib_u8 *acode, const mlib_u8
    *ucode, mlib_s32 n);
```

DESCRIPTION The `mlib_SignaluLaw2ALaw()` function performs ITU G.711 m-law and A-law compression and decompression in compliance with the ITU (formerly CCITT) G.711 specification.

PARAMETERS The function takes the following arguments:

acode A-law code array.

ucode m-law code array.

n Number of samples in the input array.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mlib_SignalALaw2Linear(3MLIB)`, `mlib_SignalALaw2uLaw(3MLIB)`, `mlib_SignalLinear2ALaw(3MLIB)`, `mlib_SignalLinear2uLaw(3MLIB)`, `mlib_SignaluLaw2Linear(3MLIB)`, `attributes(5)`

mllib_SignaluLaw2Linear(3MLIB)

NAME mllib_SignaluLaw2Linear – ITU G.711 m-law and A-law compression and decompression

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignaluLaw2Linear(mllib_s16 *pcm, const mllib_u8
    *ucode, mllib_s32 n);
```

DESCRIPTION The `mllib_SignaluLaw2Linear()` function performs ITU G.711 m-law and A-law compression and decompression in compliance with the ITU (formerly CCITT) G.711 specification.

PARAMETERS The function takes the following arguments:

pcm Linear PCM sample array.

ucode m-law code array.

n Number of samples in the input array.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalALaw2Linear(3MLIB)`, `mllib_SignalALaw2uLaw(3MLIB)`, `mllib_SignalLinear2ALaw(3MLIB)`, `mllib_SignalLinear2uLaw(3MLIB)`, `mllib_SignaluLaw2ALaw(3MLIB)`, `attributes(5)`

mlib_SignalUpSampleFIR_F32_F32(3MLIB)

NAME | mlib_SignalUpSampleFIR_F32_F32 – upsampling with filtering

SYNOPSIS |

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_SignalUpSampleFIR_F32_F32(mlib_f32 *dst, const
mlib_f32 *src, void *state, mlib_s32 n);
```

DESCRIPTION | The `mlib_SignalUpSampleFIR_F32_F32()` function performs upsampling immediately followed by FIR filtering on one packet of signal and updates the internal state.

PARAMETERS | The function takes the following arguments:

dst | Output signal array.

src | Input signal array.

state | Internal state structure.

n | Number of samples in the input signal array.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `attributes(5)`

mllib_SignalUpSampleFIR_F32S_F32S(3MLIB)

NAME | mllib_SignalUpSampleFIR_F32S_F32S – upsampling with filtering

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalUpSampleFIR_F32S_F32S(mllib_f32 *dst, const
mllib_f32 *src, void *state, mllib_s32 n);
```

DESCRIPTION | The `mllib_SignalUpSampleFIR_F32S_F32S()` function performs upsampling immediately followed by FIR filtering on one packet of signal and updates the internal state.

PARAMETERS | The function takes the following arguments:

dst | Output stereo signal array. `src[2*i]` contains Channel 0, and `src[2*i+1]` contains Channel 1.

src | Source stereo signal array. `src[2*i]` contains Channel 0, and `src[2*i+1]` contains Channel 1.

state | Internal state structure.

n | Number of samples in the input signal array.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `attributes(5)`

mllib_SignalUpSampleFIRFree_F32_F32(3MLIB)

NAME mllib_SignalUpSampleFIRFree_F32_F32 – upsampling with filtering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_SignalUpSampleFIRFree_F32_F32(void *state);
```

DESCRIPTION The mllib_SignalUpSampleFIRFree_F32_F32() function releases the memory allocated for the internal state structure for upsampling immediately followed by FIR filtering.

PARAMETERS The function takes the following arguments:

state Internal state structure.

RETURN VALUES None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO attributes(5)

mllib_SignalUpSampleFIRFree_F32S_F32S(3MLIB)

NAME | mllib_SignalUpSampleFIRFree_F32S_F32S – upsampling with filtering

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
void mllib_SignalUpSampleFIRFree_F32S_F32S(void *state);
```

DESCRIPTION | The `mllib_SignalUpSampleFIRFree_F32S_F32S()` function releases the memory allocated for the internal state structure for upsampling immediately followed by FIR filtering.

PARAMETERS | The function takes the following arguments:
state Internal state structure.

RETURN VALUES | None.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `attributes(5)`

mllib_SignalUpSampleFIRFree_S16_S16(3MLIB)

NAME mllib_SignalUpSampleFIRFree_S16_S16, mllib_SignalUpSampleFIRFree_S16S_S16S – upsampling with filtering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_SignalUpSampleFIRFree_S16_S16(void *state);
void mllib_SignalUpSampleFIRFree_S16S_S16S(void *state);
```

DESCRIPTION Each of these functions releases the memory allocated for the internal state structure for upsampling immediately followed by FIR filtering.

PARAMETERS Each of the functions takes the following arguments:
state Internal state structure.

RETURN VALUES None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_SignalUpSampleFIR_S16_S16_Sat(3MLIB),
mllib_SignalUpSampleFIRInit_S16_S16(3MLIB), attributes(5)

mllib_SignalUpSampleFIRInit_F32_F32(3MLIB)

NAME mllib_SignalUpSampleFIRInit_F32_F32 – upsampling with filtering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalUpSampleFIRInit_F32_F32(void **state, const
mllib_f32 *flt, mllib_s32 tap, mllib_s32 factor, mllib_s32 phase);
```

DESCRIPTION The mllib_SignalUpSampleFIRInit_F32_F32() function allocates memory for the internal state structure and converts the parameters into an internal representation for upsampling immediately followed by FIR filtering.

PARAMETERS The function takes the following arguments:

<i>state</i>	Internal state structure.
<i>flt</i>	Filter coefficient array.
<i>tap</i>	Taps of the filter.
<i>factor</i>	Factor by which to upsample.
<i>phase</i>	Parameter that determines the relative position of an input value, within the output signal. $0 \leq \text{phase} < \text{factor}$.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO attributes(5)

mllib_SignalUpSampleFIRInit_F32S_F32S(3MLIB)

NAME mllib_SignalUpSampleFIRInit_F32S_F32S – upsampling with filtering

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalUpSampleFIRInit_F32S_F32S(void **state,
    const mllib_f32 *flt, mllib_s32 tap, mllib_s32 factor, mllib_s32
    phase) ;
```

DESCRIPTION The `mllib_SignalUpSampleFIRInit_F32S_F32S()` function allocates memory for the internal state structure and converts the parameters into an internal representation for upsampling immediately followed by FIR filtering.

PARAMETERS The function takes the following arguments:

<i>state</i>	Internal state structure.
<i>flt</i>	Filter coefficient array in two-channel stereo format. <code>src[2*i]</code> contains channel 0, and <code>src[2*i+1]</code> contains channel 1 array.
<i>tap</i>	Taps of the filter.
<i>factor</i>	Factor by which to upsample.
<i>phase</i>	Parameter that determines the relative position of an input value, within the output signal. $0 \leq \text{phase} < \text{factor}$.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `attributes(5)`

mllib_SignalUpSampleFIRInit_S16_S16(3MLIB)

NAME	mllib_SignalUpSampleFIRInit_S16_S16, mllib_SignalUpSampleFIRInit_S16S_S16S – upsampling with filtering						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalUpSampleFIRInit_S16_S16(void **state, const mllib_f32 *flt, mllib_s32 tap, mllib_s32 factor, mllib_s32 phase); mllib_status mllib_SignalUpSampleFIRInit_S16S_S16S(void **state, const mllib_f32 *flt, mllib_s32 tap, mllib_s32 factor, mllib_s32 phase);</pre>						
DESCRIPTION	Each of these functions allocates memory for the internal state structure and converts the parameters into an internal representation for upsampling immediately followed by FIR filtering.						
PARAMETERS	Each of the functions takes the following arguments: <i>state</i> Internal state structure. <i>flt</i> Filter coefficient array. <i>tap</i> Taps of the filter. <i>factor</i> Factor by which to upsample. <i>phase</i> Parameter that determines the relative position of an input value, within the output signal. $0 \leq \text{phase} < \text{factor}$.						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_SignalUpSampleFIR_S16_S16_Sat(3MLIB), mllib_SignalUpSampleFIRFree_S16_S16(3MLIB), attributes(5)						

mlib_SignalUpSampleFIR_S16_S16_Sat(3MLIB)

NAME | mlib_SignalUpSampleFIR_S16_S16_Sat, mlib_SignalUpSampleFIR_S16S_S16S_Sat – upsampling with filtering

SYNOPSIS |

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_SignalUpSampleFIR_S16_S16_Sat(mlib_s16 *dst,
const mlib_s16 *src, void *state, mlib_s32 n);

mlib_status mlib_SignalUpSampleFIR_S16S_S16S_Sat(mlib_s16 *dst,
const mlib_s16 *src, void *state, mlib_s32 n);
```

DESCRIPTION | Each of these functions performs upsampling immediately followed by FIR filtering on one packet of signal and updates the internal state.

PARAMETERS | Each of the functions takes the following arguments:

dst | Output signal array.

src | Input signal array.

state | Internal state structure.

n | Number of samples in the input signal array.

RETURN VALUES | Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mlib_SignalUpSampleFIRFree_S16_S16(3MLIB),
mlib_SignalUpSampleFIRInit_S16_S16(3MLIB), attributes(5)

mllib_SignalUpSample_S16_S16(3MLIB)

NAME	mllib_SignalUpSample_S16_S16, mllib_SignalUpSample_S16S_S16S, mllib_SignalUpSample_F32_F32, mllib_SignalUpSample_F32S_F32S – signal upsampling										
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_SignalUpSample_S16_S16(mllib_s16 *dst, const mllib_s16 *src, mllib_s32 factor, mllib_s32 phase, mllib_s32 n); mllib_status mllib_SignalUpSample_S16S_S16S(mllib_s16 *dst, const mllib_s16 *src, mllib_s32 factor, mllib_s32 phase, mllib_s32 n); mllib_status mllib_SignalUpSample_F32_F32(mllib_f32 *dst, const mllib_f32 *src, mllib_s32 factor, mllib_s32 phase, mllib_s32 n); mllib_status mllib_SignalUpSample_F32S_F32S(mllib_f32 *dst, const mllib_f32 *src, mllib_s32 factor, mllib_s32 phase, mllib_s32 n);</pre>										
DESCRIPTION	<p>Each of these functions performs upsampling.</p> <p>For monaural signals, the following equation is used:</p> $\begin{aligned} \text{dst}[i] &= \text{src}[k] && \text{if } i == k * \text{factor} + \text{phase} \\ \text{dst}[i] &= 0 && \text{if } i \neq k * \text{factor} + \text{phase} \end{aligned}$ <p>where $k = 0, 1, \dots, (n - 1)$; $i = 0, 1, \dots, (n * \text{factor} - 1)$.</p> <p>For stereo signals, the following equation is used:</p> $\begin{aligned} \text{dst}[2*i] &= \text{src}[2*k] && \text{if } i == k * \text{factor} + \text{phase} \\ \text{dst}[2*i] &= 0 && \text{if } i \neq k * \text{factor} + \text{phase} \\ \text{dst}[2*i + 1] &= \text{src}[2*k + 1] && \text{if } i == k * \text{factor} + \text{phase} \\ \text{dst}[2*i + 1] &= 0 && \text{if } i \neq k * \text{factor} + \text{phase} \end{aligned}$ <p>where $k = 0, 1, \dots, (n - 1)$; $i = 0, 1, \dots, (n * \text{factor} - 1)$.</p>										
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <table><tr><td><i>dst</i></td><td>Output signal array.</td></tr><tr><td><i>src</i></td><td>Input signal array.</td></tr><tr><td><i>factor</i></td><td>Factor by which to upsample. $\text{factor} \geq 1$.</td></tr><tr><td><i>phase</i></td><td>Parameter that determines relative position of an input value, within the output signal. $0 \leq \text{phase} < \text{factor}$.</td></tr><tr><td><i>n</i></td><td>Number of samples in the input signal array.</td></tr></table>	<i>dst</i>	Output signal array.	<i>src</i>	Input signal array.	<i>factor</i>	Factor by which to upsample. $\text{factor} \geq 1$.	<i>phase</i>	Parameter that determines relative position of an input value, within the output signal. $0 \leq \text{phase} < \text{factor}$.	<i>n</i>	Number of samples in the input signal array.
<i>dst</i>	Output signal array.										
<i>src</i>	Input signal array.										
<i>factor</i>	Factor by which to upsample. $\text{factor} \geq 1$.										
<i>phase</i>	Parameter that determines relative position of an input value, within the output signal. $0 \leq \text{phase} < \text{factor}$.										
<i>n</i>	Number of samples in the input signal array.										
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.										

`mlib_SignalUpSample_S16_S16(3MLIB)`

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mlib_SignalDownSample_S16_S16(3MLIB)`, `attributes(5)`

mllib_SignalWhiteNoise_F32(3MLIB)

NAME | mllib_SignalWhiteNoise_F32 – white noise generation

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
mllib_status mllib_SignalWhiteNoise_F32(mllib_f32 *wnoise, void *state,  
mllib_s32 n);
```

DESCRIPTION | The `mllib_SignalWhiteNoise_F32()` function generates one packet of white noise and updates the internal state.

PARAMETERS | The function takes the following arguments:

<i>wnoise</i>	Generated white noise array.
<i>state</i>	Internal state structure.
<i>n</i>	Length of the generated sine wave array in number of samples.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `attributes(5)`

NAME | mllib_SignalWhiteNoiseFree_F32 – white noise generation

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_SignalWhiteNoiseFree_F32(void *state);
```

DESCRIPTION | The `mllib_SignalWhiteNoiseFree_F32()` function releases the memory allocated for the internal state's structure.

PARAMETERS | The function takes the following arguments:
state Internal state structure.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `attributes(5)`

mllib_SignalWhiteNoiseFree_S16(3MLIB)

NAME | mllib_SignalWhiteNoiseFree_S16 – white noise generation

SYNOPSIS | cc [*flag...*] *file...* -lmllib [*library...*]
#include <mllib.h>

```
void mllib_SignalWhiteNoiseFree_S16(void *state);
```

DESCRIPTION | The mllib_SignalWhiteNoiseFree_S16() function releases the memory allocated for the internal state's structure.

PARAMETERS | The function takes the following arguments:
state Internal state structure.

RETURN VALUES | None.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_SignalWhiteNoise_S16\(3MLIB\)](#),
[mllib_SignalWhiteNoiseInit_S16\(3MLIB\)](#), [attributes\(5\)](#)

NAME mllib_SignalWhiteNoiseInit_F32 – white noise generation

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalWhiteNoiseInit_F32(void **state, mllib_f32
      mag, mllib_f32 seed);
```

DESCRIPTION The `mllib_SignalWhiteNoiseInit_F32()` function allocates memory for an internal state structure and converts the parameters into an internal representation.

PARAMETERS The function takes the following arguments:

state Internal state structure.

mag Magnitude of white noise to be generated, in Q15 format.

seed Seed value for the pseudorandom number generator.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `attributes(5)`

mllib_SignalWhiteNoiseInit_S16(3MLIB)

NAME | mllib_SignalWhiteNoiseInit_S16 – white noise generation

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalWhiteNoiseInit_S16(void **state, mllib_s16
      mag, mllib_s16 seed);
```

DESCRIPTION | The mllib_SignalWhiteNoiseInit_S16() function allocates memory for an internal state structure and converts the parameters into an internal representation.

PARAMETERS | The function takes the following arguments:

state | Internal state structure.

mag | Magnitude of white noise to be generated, in Q15 format.

seed | Seed value for the pseudorandom number generator.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_SignalWhiteNoise_S16(3MLIB),
mllib_SignalWhiteNoiseFree_S16(3MLIB), attributes(5)

NAME mllib_SignalWhiteNoise_S16 – white noise generation

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_SignalWhiteNoise_S16(mllib_s16 *wnoise, void *state,
mllib_s32 n);
```

DESCRIPTION The `mllib_SignalWhiteNoise_S16()` function generates one packet of white noise and updates the internal state.

PARAMETERS The function takes the following arguments:

wnoise Generated white noise array.

state Internal state structure.

n Length of the generated sine wave array in number of samples.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_SignalWhiteNoiseFree_S16(3MLIB)`,
`mllib_SignalWhiteNoiseInit_S16(3MLIB)`, `attributes(5)`

mllib_VectorAddS_U8_Mod(3MLIB)

NAME	mllib_VectorAddS_U8_Mod, mllib_VectorAddS_U8_Sat, mllib_VectorAddS_U8C_Mod, mllib_VectorAddS_U8C_Sat, mllib_VectorAddS_S8_Mod, mllib_VectorAddS_S8_Sat, mllib_VectorAddS_S8C_Mod, mllib_VectorAddS_S8C_Sat, mllib_VectorAddS_S16_Mod, mllib_VectorAddS_S16_Sat, mllib_VectorAddS_S16C_Mod, mllib_VectorAddS_S16C_Sat, mllib_VectorAddS_S32_Mod, mllib_VectorAddS_S32_Sat, mllib_VectorAddS_S32C_Mod, mllib_VectorAddS_S32C_Sat – vector addition to scalar, in place
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VectorAddS_U8_Mod(mllib_u8 *xz, const mllib_u8 *c, mllib_s32 n); mllib_status mllib_VectorAddS_U8_Sat(mllib_u8 *xz, const mllib_u8 *c, mllib_s32 n); mllib_status mllib_VectorAddS_U8C_Mod(mllib_u8 *xz, const mllib_u8 *c, mllib_s32 n); mllib_status mllib_VectorAddS_U8C_Sat(mllib_u8 *xz, const mllib_u8 *c, mllib_s32 n); mllib_status mllib_VectorAddS_S8_Mod(mllib_s8 *xz, const mllib_s8 *c, mllib_s32 n); mllib_status mllib_VectorAddS_S8_Sat(mllib_s8 *xz, const mllib_s8 *c, mllib_s32 n); mllib_status mllib_VectorAddS_S8C_Mod(mllib_s8 *xz, const mllib_s8 *c, mllib_s32 n); mllib_status mllib_VectorAddS_S8C_Sat(mllib_s8 *xz, const mllib_s8 *c, mllib_s32 n); mllib_status mllib_VectorAddS_S16_Mod(mllib_s16 *xz, const mllib_s16 *c, mllib_s32 n); mllib_status mllib_VectorAddS_S16_Sat(mllib_s16 *xz, const mllib_s16 *c, mllib_s32 n); mllib_status mllib_VectorAddS_S16C_Mod(mllib_s16 *xz, const mllib_s16 *c, mllib_s32 n); mllib_status mllib_VectorAddS_S16C_Sat(mllib_s16 *xz, const mllib_s16 *c, mllib_s32 n); mllib_status mllib_VectorAddS_S32_Mod(mllib_s32 *xz, const mllib_s32 *c, mllib_s32 n); mllib_status mllib_VectorAddS_S32_Sat(mllib_s32 *xz, const mllib_s32 *c, mllib_s32 n); mllib_status mllib_VectorAddS_S32C_Mod(mllib_s32 *xz, const mllib_s32 *c, mllib_s32 n);</pre>

mllib_VectorAddS_U8_Mod(3MLIB)

```
mllib_status mllib_VectorAddS_S32C_Sat(mllib_s32 *xz, const mllib_s32 *c, mllib_s32 n);
```

DESCRIPTION Each of these functions performs an in-place addition of a scalar to a vector.

For real data, the following equation is used:

$$xz[i] = c[0] + xz[i]$$

where $i = 0, 1, \dots, (n - 1)$.

For complex data, the following equation is used:

$$\begin{aligned}xz[2*i] &= c[0] + xz[2*i] \\xz[2*i + 1] &= c[1] + xz[2*i + 1]\end{aligned}$$

where $i = 0, 1, \dots, (n - 1)$.

PARAMETERS Each of the functions takes the following arguments:

- xz* Pointer to the first element of the source and destination vector.
- c* Pointer to the source scalar. When the function is used with complex data types, *c*[0] contains the scalar for the real part, and *c*[1] contains the scalar for the imaginary part.
- n* Number of elements in the vectors.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_VectorAddS_U8_U8_Mod(3MLIB)`, `attributes(5)`

mllib_VectorAddS_U8_U8_Mod(3MLIB)

NAME	mllib_VectorAddS_U8_U8_Mod, mllib_VectorAddS_U8_U8_Sat, mllib_VectorAddS_U8C_U8C_Mod, mllib_VectorAddS_U8C_U8C_Sat, mllib_VectorAddS_S8_S8_Mod, mllib_VectorAddS_S8_S8_Sat, mllib_VectorAddS_S8C_S8C_Mod, mllib_VectorAddS_S8C_S8C_Sat, mllib_VectorAddS_S16_U8_Mod, mllib_VectorAddS_S16_U8_Sat, mllib_VectorAddS_S16_S8_Mod, mllib_VectorAddS_S16_S8_Sat, mllib_VectorAddS_S16_S16_Mod, mllib_VectorAddS_S16_S16_Sat, mllib_VectorAddS_S16C_U8C_Mod, mllib_VectorAddS_S16C_U8C_Sat, mllib_VectorAddS_S16C_S8C_Mod, mllib_VectorAddS_S16C_S8C_Sat, mllib_VectorAddS_S16C_S16C_Mod, mllib_VectorAddS_S16C_S16C_Sat, mllib_VectorAddS_S32_S16_Mod, mllib_VectorAddS_S32_S16_Sat, mllib_VectorAddS_S32_S32_Mod, mllib_VectorAddS_S32_S32_Sat, mllib_VectorAddS_S32C_S16C_Mod, mllib_VectorAddS_S32C_S16C_Sat, mllib_VectorAddS_S32C_S32C_Mod, mllib_VectorAddS_S32C_S32C_Sat – vector addition to scalar
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VectorAddS_U8_U8_Mod(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *c, mllib_s32 n); mllib_status mllib_VectorAddS_U8_U8_Sat(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *c, mllib_s32 n); mllib_status mllib_VectorAddS_U8C_U8C_Mod(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *c, mllib_s32 n); mllib_status mllib_VectorAddS_U8C_U8C_Sat(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *c, mllib_s32 n); mllib_status mllib_VectorAddS_S8_S8_Mod(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *c, mllib_s32 n); mllib_status mllib_VectorAddS_S8_S8_Sat(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *c, mllib_s32 n); mllib_status mllib_VectorAddS_S8C_S8C_Mod(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *c, mllib_s32 n); mllib_status mllib_VectorAddS_S8C_S8C_Sat(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *c, mllib_s32 n); mllib_status mllib_VectorAddS_S16_U8_Mod(mllib_s16 *z, const mllib_u8 *x, const mllib_u8 *c, mllib_s32 n); mllib_status mllib_VectorAddS_S16_U8_Sat(mllib_s16 *z, const mllib_u8 *x, const mllib_u8 *c, mllib_s32 n); mllib_status mllib_VectorAddS_S16_S8_Mod(mllib_s16 *z, const mllib_s8 *x, const mllib_s8 *c, mllib_s32 n); mllib_status mllib_VectorAddS_S16_S8_Sat(mllib_s16 *z, const mllib_s8 *x, const mllib_s8 *c, mllib_s32 n);</pre>

mllib_VectorAddS_U8_U8_Mod(3MLIB)

```
mllib_status mllib_VectorAddS_S16_S16_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 n);
mllib_status mllib_VectorAddS_S16_S16_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 n);
mllib_status mllib_VectorAddS_S16C_U8C_Mod(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *c, mllib_s32 n);
mllib_status mllib_VectorAddS_S16C_U8C_Sat(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *c, mllib_s32 n);
mllib_status mllib_VectorAddS_S16C_S8C_Mod(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *c, mllib_s32 n);
mllib_status mllib_VectorAddS_S16C_S8C_Sat(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *c, mllib_s32 n);
mllib_status mllib_VectorAddS_S16C_S16C_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 n);
mllib_status mllib_VectorAddS_S16C_S16C_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 n);
mllib_status mllib_VectorAddS_S32_S16_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 n);
mllib_status mllib_VectorAddS_S32_S16_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 n);
mllib_status mllib_VectorAddS_S32_S32_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *c, mllib_s32 n);
mllib_status mllib_VectorAddS_S32_S32_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *c, mllib_s32 n);
mllib_status mllib_VectorAddS_S32C_S16C_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 n);
mllib_status mllib_VectorAddS_S32C_S16C_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 n);
mllib_status mllib_VectorAddS_S32C_S32C_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *c, mllib_s32 n);
mllib_status mllib_VectorAddS_S32C_S32C_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *c, mllib_s32 n);
```

DESCRIPTION

Each of these functions adds a scalar to a vector.

For real data, the following equation is used:

$$z[i] = c[0] + x[i]$$

where $i = 0, 1, \dots, (n - 1)$.

For complex data, the following equation is used:

mllib_VectorAddS_U8_U8_Mod(3MLIB)

```
z[2*i]      = c[0] + x[2*i]
z[2*i + 1] = c[1] + x[2*i + 1]
```

where $i = 0, 1, \dots, (n - 1)$.

PARAMETERS Each of the functions takes the following arguments:

z Pointer to the first element of the destination vector.
x Pointer to the first element of the source vector.
c Pointer to the source scalar. When the function is used with complex data types, *c*[0] contains the scalar for the real part, and *c*[1] contains the scalar for the imaginary part.
n Number of elements in the vectors.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_VectorAddS_U8_Mod(3MLIB)`, `attributes(5)`

NAME	mllib_VectorAdd_U8_Mod, mllib_VectorAdd_U8_Sat, mllib_VectorAdd_U8C_Mod, mllib_VectorAdd_U8C_Sat, mllib_VectorAdd_S8_Mod, mllib_VectorAdd_S8_Sat, mllib_VectorAdd_S8C_Mod, mllib_VectorAdd_S8C_Sat, mllib_VectorAdd_S16_Mod, mllib_VectorAdd_S16_Sat, mllib_VectorAdd_S16C_Mod, mllib_VectorAdd_S16C_Sat, mllib_VectorAdd_S32_Mod, mllib_VectorAdd_S32_Sat, mllib_VectorAdd_S32C_Mod, mllib_VectorAdd_S32C_Sat – vector addition, in place
SYNOPSIS	<pre>cc [flag...] file... -lmlib [library...] #include <mllib.h> mllib_status mllib_VectorAdd_U8_Mod(mllib_u8 *xz, const mllib_u8 *y, mllib_s32 n); mllib_status mllib_VectorAdd_U8_Sat(mllib_u8 *xz, const mllib_u8 *y, mllib_s32 n); mllib_status mllib_VectorAdd_U8C_Mod(mllib_u8 *xz, const mllib_u8 *y, mllib_s32 n); mllib_status mllib_VectorAdd_U8C_Sat(mllib_u8 *xz, const mllib_u8 *y, mllib_s32 n); mllib_status mllib_VectorAdd_S8_Mod(mllib_s8 *xz, const mllib_s8 *y, mllib_s32 n); mllib_status mllib_VectorAdd_S8_Sat(mllib_s8 *xz, const mllib_s8 *y, mllib_s32 n); mllib_status mllib_VectorAdd_S8C_Mod(mllib_s8 *xz, const mllib_s8 *y, mllib_s32 n); mllib_status mllib_VectorAdd_S8C_Sat(mllib_s8 *xz, const mllib_s8 *y, mllib_s32 n); mllib_status mllib_VectorAdd_S16_Mod(mllib_s16 *xz, const mllib_s16 *y, mllib_s32 n); mllib_status mllib_VectorAdd_S16_Sat(mllib_s16 *xz, const mllib_s16 *y, mllib_s32 n); mllib_status mllib_VectorAdd_S16C_Mod(mllib_s16 *xz, const mllib_s16 *y, mllib_s32 n); mllib_status mllib_VectorAdd_S16C_Sat(mllib_s16 *xz, const mllib_s16 *y, mllib_s32 n); mllib_status mllib_VectorAdd_S32_Mod(mllib_s32 *xz, const mllib_s32 *y, mllib_s32 n); mllib_status mllib_VectorAdd_S32_Sat(mllib_s32 *xz, const mllib_s32 *y, mllib_s32 n); mllib_status mllib_VectorAdd_S32C_Mod(mllib_s32 *xz, const mllib_s32 *y, mllib_s32 n); mllib_status mllib_VectorAdd_S32C_Sat(mllib_s32 *xz, const mllib_s32 *y, mllib_s32 n);</pre>

mllib_VectorAdd_U8_Mod(3MLIB)

DESCRIPTION Each of these functions performs the in-place addition of one vector to another vector.

It uses the following equation:

$$xz[i] = xz[i] + y[i]$$

where $i = 0, 1, \dots, (n - 1)$ for real data; $i = 0, 1, \dots, (2*n - 1)$ for complex data.

PARAMETERS Each of the functions takes the following arguments:

xz Pointer to the first element of the first source and destination vector.

y Pointer to the first element of the second source vector.

n Number of elements in the vectors.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_VectorAdd_U8_U8_Mod\(3MLIB\)](#), `attributes(5)`

mllib_VectorAdd_U8_U8_Mod(3MLIB)

NAME | mllib_VectorAdd_U8_U8_Mod, mllib_VectorAdd_U8_U8_Sat,
 mllib_VectorAdd_U8C_U8C_Mod, mllib_VectorAdd_U8C_U8C_Sat,
 mllib_VectorAdd_S8_S8_Mod, mllib_VectorAdd_S8_S8_Sat,
 mllib_VectorAdd_S8C_S8C_Mod, mllib_VectorAdd_S8C_S8C_Sat,
 mllib_VectorAdd_S16_U8_Mod, mllib_VectorAdd_S16_U8_Sat,
 mllib_VectorAdd_S16_S8_Mod, mllib_VectorAdd_S16_S8_Sat,
 mllib_VectorAdd_S16_S16_Mod, mllib_VectorAdd_S16_S16_Sat,
 mllib_VectorAdd_S16C_U8C_Mod, mllib_VectorAdd_S16C_U8C_Sat,
 mllib_VectorAdd_S16C_S8C_Mod, mllib_VectorAdd_S16C_S8C_Sat,
 mllib_VectorAdd_S16C_S16C_Mod, mllib_VectorAdd_S16C_S16C_Sat,
 mllib_VectorAdd_S32_S16_Mod, mllib_VectorAdd_S32_S16_Sat,
 mllib_VectorAdd_S32_S32_Mod, mllib_VectorAdd_S32_S32_Sat,
 mllib_VectorAdd_S32C_S16C_Mod, mllib_VectorAdd_S32C_S16C_Sat,
 mllib_VectorAdd_S32C_S32C_Mod, mllib_VectorAdd_S32C_S32C_Sat – vector addition

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VectorAdd_U8_U8_Mod(mllib_u8 *z, const mllib_u8
    *x, const mllib_u8 *y, mllib_s32 n);

mllib_status mllib_VectorAdd_U8_U8_Sat(mllib_u8 *z, const mllib_u8
    *x, const mllib_u8 *y, mllib_s32 n);

mllib_status mllib_VectorAdd_U8C_U8C_Mod(mllib_u8 *z, const mllib_u8
    *x, const mllib_u8 *y, mllib_s32 n);

mllib_status mllib_VectorAdd_U8C_U8C_Sat(mllib_u8 *z, const mllib_u8
    *x, const mllib_u8 *y, mllib_s32 n);

mllib_status mllib_VectorAdd_S8_S8_Mod(mllib_s8 *z, const mllib_s8
    *x, const mllib_s8 *y, mllib_s32 n);

mllib_status mllib_VectorAdd_S8_S8_Sat(mllib_s8 *z, const mllib_s8
    *x, const mllib_s8 *y, mllib_s32 n);

mllib_status mllib_VectorAdd_S8C_S8C_Mod(mllib_s8 *z, const mllib_s8
    *x, const mllib_s8 *y, mllib_s32 n);

mllib_status mllib_VectorAdd_S8C_S8C_Sat(mllib_s8 *z, const mllib_s8
    *x, const mllib_s8 *y, mllib_s32 n);

mllib_status mllib_VectorAdd_S16_U8_Mod(mllib_s16 *z, const mllib_u8
    *x, const mllib_u8 *y, mllib_s32 n);

mllib_status mllib_VectorAdd_S16_U8_Sat(mllib_s16 *z, const mllib_u8
    *x, const mllib_u8 *y, mllib_s32 n);

mllib_status mllib_VectorAdd_S16_S8_Mod(mllib_s16 *z, const mllib_s8
    *x, const mllib_s8 *y, mllib_s32 n);

mllib_status mllib_VectorAdd_S16_S8_Sat(mllib_s16 *z, const mllib_s8
    *x, const mllib_s8 *y, mllib_s32 n);

mllib_status mllib_VectorAdd_S16_S16_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 n);
```

mllib_VectorAdd_U8_U8_Mod(3MLIB)

```
mllib_status mllib_VectorAdd_S16_S16_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 n);
mllib_status mllib_VectorAdd_S16C_U8C_Mod(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *y, mllib_s32 n);
mllib_status mllib_VectorAdd_S16C_U8C_Sat(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *y, mllib_s32 n);
mllib_status mllib_VectorAdd_S16C_S8C_Mod(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *y, mllib_s32 n);
mllib_status mllib_VectorAdd_S16C_S8C_Sat(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *y, mllib_s32 n);
mllib_status mllib_VectorAdd_S16C_S16C_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 n);
mllib_status mllib_VectorAdd_S16C_S16C_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 n);
mllib_status mllib_VectorAdd_S32_S16_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 n);
mllib_status mllib_VectorAdd_S32_S16_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 n);
mllib_status mllib_VectorAdd_S32_S32_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 n);
mllib_status mllib_VectorAdd_S32_S32_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 n);
mllib_status mllib_VectorAdd_S32C_S16C_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 n);
mllib_status mllib_VectorAdd_S32C_S16C_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 n);
mllib_status mllib_VectorAdd_S32C_S32C_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 n);
mllib_status mllib_VectorAdd_S32C_S32C_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 n);
```

DESCRIPTION	Each of these functions performs the addition of one vector to another vector. It uses the following equation: $z[i] = x[i] + y[i]$ where $i = 0, 1, \dots, (n - 1)$ for real data; $i = 0, 1, \dots, (2*n - 1)$ for complex data.
PARAMETERS	Each of the functions takes the following arguments: z Pointer to the first element of the destination vector.

`mllib_VectorAdd_U8_U8_Mod(3MLIB)`

x Pointer to the first element of the first source vector.
y Pointer to the first element of the second source vector.
n Number of elements in the vectors.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_VectorAdd_U8_Mod(3MLIB)`, `attributes(5)`

mllib_VectorAng_U8C(3MLIB)

NAME	mllib_VectorAng_U8C, mllib_VectorAng_S8C, mllib_VectorAng_S16C, mllib_VectorAng_S32C – vector complex phase (angle)						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VectorAng_U8C(mllib_d64 *a, const mllib_u8 *x, mllib_s32 n); mllib_status mllib_VectorAng_S8C(mllib_d64 *a, const mllib_s8 *x, mllib_s32 n); mllib_status mllib_VectorAng_S16C(mllib_d64 *a, const mllib_s16 *x, mllib_s32 n); mllib_status mllib_VectorAng_S32C(mllib_d64 *a, const mllib_s32 *x, mllib_s32 n);</pre>						
DESCRIPTION	<p>Each of these functions computes the phase vector of a complex vector.</p> <p>The following equation is used:</p> $a[i] = \text{atan}(x[2*i + 1] / x[2*i])$ <p>where $i = 0, 1, \dots, (n - 1)$.</p>						
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <p><i>a</i> Pointer to the destination phase vector.</p> <p><i>x</i> Pointer to the source vector</p> <p><i>n</i> Number of elements in the vector.</p>						
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	attributes(5)						

mlib_VectorConjRev_S8C_S8C_Sat(3MLIB)

NAME mlib_VectorConjRev_S8C_S8C_Sat, mlib_VectorConjRev_S16C_S16C_Sat, mlib_VectorConjRev_S32C_S32C_Sat – vector conjugation reversion

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_VectorConjRev_S8C_S8C_Sat(mlib_s8 *z, const
mlib_s8 *x, mlib_s32 n);

mlib_status mlib_VectorConjRev_S16C_S16C_Sat(mlib_s16 *z, const
mlib_s16 *x, mlib_s32 n);

mlib_status mlib_VectorConjRev_S32C_S32C_Sat(mlib_s32 *z, const
mlib_s32 *x, mlib_s32 n);
```

DESCRIPTION Each of these functions computes the complex reversion of a complex vector.

The source and destination vectors must be in the same data type.

The following equation is used:

$$z[2*i] = x[2*(n - 1 - i)]$$

$$z[2*i + 1] = -x[2*(n - 1 - i) + 1]$$

where $i = 0, 1, \dots, (n - 1)$.

PARAMETERS Each of the functions takes the following arguments:

z Pointer to the first element of the destination vector.

x Pointer to the first element of the source vector.

n Number of elements in the vectors.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO attributes(5)

mllib_VectorConj_S8C_S8C_Sat(3MLIB)

NAME	mllib_VectorConj_S8C_S8C_Sat, mllib_VectorConj_S16C_S16C_Sat, mllib_VectorConj_S32C_S32C_Sat – vector conjugation						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VectorConj_S8C_S8C_Sat(mllib_s8 *z, const mllib_s8 *x, mllib_s32 n); mllib_status mllib_VectorConj_S16C_S16C_Sat(mllib_s16 *z, const mllib_s16 *x, mllib_s32 n); mllib_status mllib_VectorConj_S32C_S32C_Sat(mllib_s32 *z, const mllib_s32 *x, mllib_s32 n);</pre>						
DESCRIPTION	<p>Each of these functions computes the complex conjugate of a complex vector.</p> <p>The source and destination vectors must be in the same data type.</p> <p>The following equation is used:</p> $\begin{aligned}z[2*i] &= x[2*i] \\z[2*i + 1] &= -x[2*i + 1]\end{aligned}$ <p>where $i = 0, 1, \dots, (n - 1)$.</p>						
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <p><i>z</i> Pointer to the first element of the destination vector.</p> <p><i>x</i> Pointer to the first element of the source vector.</p> <p><i>n</i> Number of elements in the vectors.</p>						
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	attributes(5)						

NAME mllib_VectorConj_S8C_Sat, mllib_VectorConj_S16C_Sat, mllib_VectorConj_S32C_Sat – vector conjugation

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VectorConj_S8C_Sat(mllib_s8 *xz, mllib_s32 n);
mllib_status mllib_VectorConj_S16C_Sat(mllib_s16 *xz, mllib_s32 n);
mllib_status mllib_VectorConj_S32C_Sat(mllib_s32 *xz, mllib_s32 n);
```

DESCRIPTION Each of these functions computes the in-place complex conjugate of a complex vector. The following equation is used:

$$xz[2*i + 1] = -xz[2*i + 1]$$
where $i = 0, 1, \dots, (n - 1)$.

PARAMETERS Each of the functions takes the following arguments:
xz Pointer to the first element of the source and destination vector.
n Number of elements in the vector.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO attributes(5)

mllib_VectorConjSymExt_S8C_S8C_Sat(3MLIB)

NAME	mllib_VectorConjSymExt_S8C_S8C_Sat, mllib_VectorConjSymExt_S16C_S16C_Sat, mllib_VectorConjSymExt_S32C_S32C_Sat – vector conjugate-symmetric extension
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_VectorConjSymExt_S8C_S8C_Sat(mllib_s8 *z, const mllib_s8 *x, mllib_s32 n); mllib_status mllib_VectorConjSymExt_S16C_S16C_Sat(mllib_s16 *z, const mllib_s16 *x, mllib_s32 n); mllib_status mllib_VectorConjSymExt_S32C_S32C_Sat(mllib_s32 *z, const mllib_s32 *x, mllib_s32 n);</pre>
DESCRIPTION	<p>Each of these functions computes the complex conjugate extension of a complex vector.</p> <p>The source and destination vectors must be in the same data type.</p> <p>When n is even, the following equation is used:</p> $\begin{aligned} z[2*i] &= x[2*i] \\ z[2*i + 1] &= -x[2*i + 1] \end{aligned}$ <p>for $i = 0, 1, \dots, (n - 1)$.</p> $\begin{aligned} z[2*i] &= x[2*(2*n - 1 - i)] \\ z[2*i + 1] &= -x[2*(2*n - 1 - i) + 1] \end{aligned}$ <p>for $i = n, (n + 1), \dots, (2*n - 1)$.</p> <p>When n is odd, the following equation is used:</p> $\begin{aligned} z[2*i] &= x[2*i] \\ z[2*i + 1] &= -x[2*i + 1] \end{aligned}$ <p>for $i = 0, 1, \dots, (n - 1)$.</p> $\begin{aligned} z[2*i] &= x[2*(2*n - 2 - i)] \\ z[2*i + 1] &= -x[2*(2*n - 2 - i) + 1] \end{aligned}$ <p>for $i = n, (n + 1), \dots, (2*n - 2)$.</p>
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <p>z Pointer to the first element of the destination vector.</p> <p>x Pointer to the first element of the source vector.</p> <p>n Number of elements in the source vector.</p>
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `attributes(5)`

mllib_VectorConvert_U8_S8_Mod(3MLIB)

NAME	mllib_VectorConvert_U8_S8_Mod, mllib_VectorConvert_U8_S16_Mod, mllib_VectorConvert_U8_S32_Mod, mllib_VectorConvert_S8_U8_Mod, mllib_VectorConvert_S8_S16_Mod, mllib_VectorConvert_S8_S32_Mod, mllib_VectorConvert_S16_U8_Mod, mllib_VectorConvert_S16_S8_Mod, mllib_VectorConvert_S16_S32_Mod, mllib_VectorConvert_S32_U8_Mod, mllib_VectorConvert_S32_S8_Mod, mllib_VectorConvert_S32_S16_Mod, mllib_VectorConvert_U8C_S8C_Mod, mllib_VectorConvert_U8C_S16C_Mod, mllib_VectorConvert_U8C_S32C_Mod, mllib_VectorConvert_S8C_U8C_Mod, mllib_VectorConvert_S8C_S16C_Mod, mllib_VectorConvert_S8C_S32C_Mod, mllib_VectorConvert_S16C_U8C_Mod, mllib_VectorConvert_S16C_S8C_Mod, mllib_VectorConvert_S16C_S32C_Mod, mllib_VectorConvert_S32C_U8C_Mod, mllib_VectorConvert_S32C_S8C_Mod, mllib_VectorConvert_S32C_S16C_Mod, mllib_VectorConvert_U8_S8_Sat, mllib_VectorConvert_U8_S16_Sat, mllib_VectorConvert_U8_S32_Sat, mllib_VectorConvert_S8_U8_Sat, mllib_VectorConvert_S8_S16_Sat, mllib_VectorConvert_S8_S32_Sat, mllib_VectorConvert_S16_U8_Sat, mllib_VectorConvert_S16_S8_Sat, mllib_VectorConvert_S16_S32_Sat, mllib_VectorConvert_S32_U8_Sat, mllib_VectorConvert_S32_S8_Sat, mllib_VectorConvert_S32_S16_Sat, mllib_VectorConvert_U8C_S8C_Sat, mllib_VectorConvert_U8C_S16C_Sat, mllib_VectorConvert_U8C_S32C_Sat, mllib_VectorConvert_S8C_U8C_Sat, mllib_VectorConvert_S8C_S16C_Sat, mllib_VectorConvert_S8C_S32C_Sat, mllib_VectorConvert_S16C_U8C_Sat, mllib_VectorConvert_S16C_S8C_Sat, mllib_VectorConvert_S16C_S32C_Sat, mllib_VectorConvert_S32C_U8C_Sat, mllib_VectorConvert_S32C_S8C_Sat, mllib_VectorConvert_S32C_S16C_Sat – vector data type convert
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VectorConvert_U8_S8_Mod(mllib_u8 *z, const mllib_s8 *x, mllib_s32 n); mllib_status mllib_VectorConvert_U8_S16_Mod(mllib_u8 *z, const mllib_s16 *x, mllib_s32 n); mllib_status mllib_VectorConvert_U8_S32_Mod(mllib_u8 *z, const mllib_s32 *x, mllib_s32 n); mllib_status mllib_VectorConvert_S8_U8_Mod(mllib_s8 *z, const mllib_u8 *x, mllib_s32 n); mllib_status mllib_VectorConvert_S8_S16_Mod(mllib_s8 *z, const mllib_s16 *x, mllib_s32 n); mllib_status mllib_VectorConvert_S8_S32_Mod(mllib_s8 *z, const mllib_s32 *x, mllib_s32 n); mllib_status mllib_VectorConvert_S16_U8_Mod(mllib_s16 *z, const mllib_u8 *x, mllib_s32 n); mllib_status mllib_VectorConvert_S16_S8_Mod(mllib_s16 *z, const mllib_s8 *x, mllib_s32 n);</pre>

mllib_VectorConvert_U8_S8_Mod(3MLIB)

```
mllib_status mllib_VectorConvert_S16_S32_Mod(mllib_s16 *z, const
    mllib_s32 *x, mllib_s32 n);
mllib_status mllib_VectorConvert_S32_U8_Mod(mllib_s32 *z, const
    mllib_u8 *x, mllib_s32 n);
mllib_status mllib_VectorConvert_S32_S8_Mod(mllib_s32 *z, const
    mllib_s8 *x, mllib_s32 n);
mllib_status mllib_VectorConvert_S32_S16_Mod(mllib_s32 *z, const
    mllib_s16 *x, mllib_s32 n);
mllib_status mllib_VectorConvert_U8C_S8C_Mod(mllib_u8 *z, const
    mllib_s8 *x, mllib_s32 n);
mllib_status mllib_VectorConvert_U8C_S16C_Mod(mllib_u8 *z, const
    mllib_s16 *x, mllib_s32 n);
mllib_status mllib_VectorConvert_U8C_S32C_Mod(mllib_u8 *z, const
    mllib_s32 *x, mllib_s32 n);
mllib_status mllib_VectorConvert_S8C_U8C_Mod(mllib_s8 *z, const
    mllib_u8 *x, mllib_s32 n);
mllib_status mllib_VectorConvert_S8C_S16C_Mod(mllib_s8 *z, const
    mllib_s16 *x, mllib_s32 n);
mllib_status mllib_VectorConvert_S8C_S32C_Mod(mllib_s8 *z, const
    mllib_s32 *x, mllib_s32 n);
mllib_status mllib_VectorConvert_S16C_U8C_Mod(mllib_s16 *z, const
    mllib_u8 *x, mllib_s32 n);
mllib_status mllib_VectorConvert_S16C_S8C_Mod(mllib_s16 *z, const
    mllib_s8 *x, mllib_s32 n);
mllib_status mllib_VectorConvert_S16C_S32C_Mod(mllib_s16 *z, const
    mllib_s32 *x, mllib_s32 n);
mllib_status mllib_VectorConvert_S32C_U8C_Mod(mllib_s32 *z, const
    mllib_u8 *x, mllib_s32 n);
mllib_status mllib_VectorConvert_S32C_S8C_Mod(mllib_s32 *z, const
    mllib_s8 *x, mllib_s32 n);
mllib_status mllib_VectorConvert_S32C_S16C_Mod(mllib_s32 *z, const
    mllib_s16 *x, mllib_s32 n);
mllib_status mllib_VectorConvert_U8_S8_Sat(mllib_u8 *z, const
    mllib_s8 *x, mllib_s32 n);
mllib_status mllib_VectorConvert_U8_S16_Sat(mllib_u8 *z, const
    mllib_s16 *x, mllib_s32 n);
mllib_status mllib_VectorConvert_U8_S32_Sat(mllib_u8 *z, const
    mllib_s32 *x, mllib_s32 n);
```

mllib_VectorConvert_U8_S8_Mod(3MLIB)

```
mllib_status mllib_VectorConvert_S8_U8_Sat(mllib_s8 *z, const
    mllib_u8 *x, mllib_s32 n);

mllib_status mllib_VectorConvert_S8_S16_Sat(mllib_s8 *z, const
    mllib_s16 *x, mllib_s32 n);

mllib_status mllib_VectorConvert_S8_S32_Sat(mllib_s8 *z, const
    mllib_s32 *x, mllib_s32 n);

mllib_status mllib_VectorConvert_S16_U8_Sat(mllib_s16 *z, const
    mllib_u8 *x, mllib_s32 n);

mllib_status mllib_VectorConvert_S16_S8_Sat(mllib_s16 *z, const
    mllib_s8 *x, mllib_s32 n);

mllib_status mllib_VectorConvert_S16_S32_Sat(mllib_s16 *z, const
    mllib_s32 *x, mllib_s32 n);

mllib_status mllib_VectorConvert_S32_U8_Sat(mllib_s32 *z, const
    mllib_u8 *x, mllib_s32 n);

mllib_status mllib_VectorConvert_S32_S8_Sat(mllib_s32 *z, const
    mllib_s8 *x, mllib_s32 n);

mllib_status mllib_VectorConvert_S32_S16_Sat(mllib_s32 *z, const
    mllib_s16 *x, mllib_s32 n);

mllib_status mllib_VectorConvert_U8C_S8C_Sat(mllib_u8 *z, const
    mllib_s8 *x, mllib_s32 n);

mllib_status mllib_VectorConvert_U8C_S16C_Sat(mllib_u8 *z, const
    mllib_s16 *x, mllib_s32 n);

mllib_status mllib_VectorConvert_U8C_S32C_Sat(mllib_u8 *z, const
    mllib_s32 *x, mllib_s32 n);

mllib_status mllib_VectorConvert_S8C_U8C_Sat(mllib_s8 *z, const
    mllib_u8 *x, mllib_s32 n);

mllib_status mllib_VectorConvert_S8C_S16C_Sat(mllib_s8 *z, const
    mllib_s16 *x, mllib_s32 n);

mllib_status mllib_VectorConvert_S8C_S32C_Sat(mllib_s8 *z, const
    mllib_s32 *x, mllib_s32 n);

mllib_status mllib_VectorConvert_S16C_U8C_Sat(mllib_s16 *z, const
    mllib_u8 *x, mllib_s32 n);

mllib_status mllib_VectorConvert_S16C_S8C_Sat(mllib_s16 *z, const
    mllib_s8 *x, mllib_s32 n);

mllib_status mllib_VectorConvert_S16C_S32C_Sat(mllib_s16 *z, const
    mllib_s32 *x, mllib_s32 n);

mllib_status mllib_VectorConvert_S32C_U8C_Sat(mllib_s32 *z, const
    mllib_u8 *x, mllib_s32 n);
```

mllib_VectorConvert_U8_S8_Mod(3MLIB)

```
mllib_status mllib_VectorConvert_S32C_S8C_Sat(mllib_s32 *z, const
mllib_s8 *x, mllib_s32 n);
mllib_status mllib_VectorConvert_S32C_S16C_Sat(mllib_s32 *z, const
mllib_s16 *x, mllib_s32 n);
```

DESCRIPTION

Each of these functions copies data from one vector to another vector, of different data types.

For real data, the following equation is used:

$$z[i] = x[i]$$

where $i = 0, 1, \dots, (n - 1)$.

For complex data, the following equation is used:

$$z[2*i] = x[2*i]$$

$$z[2*i + 1] = x[2*i + 1]$$

where $i = 0, 1, \dots, (n - 1)$.

See the following tables for available variations of the data type convert function.

Type [*]	U8	S8	S16	S32
U8		Y	Y	Y
S8	Y		Y	Y
S16	Y	Y		Y
S32	Y	Y	Y	

Type [*]	U8C	S8C	S16C	S32C
U8C		Y	Y	Y
S8C	Y		Y	Y
S16C	Y	Y		Y
S32C	Y	Y	Y	

[*] Each row represents a source data type. Each column represents a destination data type.

PARAMETERS

Each of the functions takes the following arguments:

- z* Pointer to the first element of the destination vector.
- x* Pointer to the first element of the source vector.
- n* Number of elements in the vectors.

mllib_VectorConvert_U8_S8_Mod(3MLIB)

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `attributes(5)`

NAME	mlib_VectorCopy_U8, mlib_VectorCopy_U8C, mlib_VectorCopy_S8, mlib_VectorCopy_S8C, mlib_VectorCopy_S16, mlib_VectorCopy_S16C, mlib_VectorCopy_S32, mlib_VectorCopy_S32C – vector copy
SYNOPSIS	<pre>cc [flag...] file... -lmlib [library...] #include <mlib.h> mlib_status mlib_VectorCopy_U8(mlib_u8 *z, const mlib_u8 *x, mlib_s32 n); mlib_status mlib_VectorCopy_U8C(mlib_u8 *z, const mlib_u8 *x, mlib_s32 n); mlib_status mlib_VectorCopy_S8(mlib_s8 *z, const mlib_s8 *x, mlib_s32 n); mlib_status mlib_VectorCopy_S8C(mlib_s8 *z, const mlib_s8 *x, mlib_s32 n); mlib_status mlib_VectorCopy_S16(mlib_s16 *z, const mlib_s16 *x, mlib_s32 n); mlib_status mlib_VectorCopy_S16C(mlib_s16 *z, const mlib_s16 *x, mlib_s32 n); mlib_status mlib_VectorCopy_S32(mlib_s32 *z, const mlib_s32 *x, mlib_s32 n); mlib_status mlib_VectorCopy_S32C(mlib_s32 *z, const mlib_s32 *x, mlib_s32 n);</pre>
DESCRIPTION	<p>Each of these functions copies one vector to another vector of the same data type.</p> <p>The input and output vectors must be in the same data type.</p> <p>For real data, the following equation is used:</p> $z[i] = x[i]$ <p>where $i = 0, 1, \dots, (n - 1)$.</p> <p>For complex data, the following equation is used:</p> $\begin{aligned} z[2*i] &= x[2*i] \\ z[2*i + 1] &= x[2*i + 1] \end{aligned}$ <p>where $i = 0, 1, \dots, (n - 1)$.</p>
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <p>z Pointer to the first element of the destination vector.</p> <p>x Pointer to the first element of the source vector.</p> <p>n Number of elements in the vectors.</p>

mllib_VectorCopy_U8(3MLIB)

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `attributes(5)`

NAME mllib_VectorDistance_U8_Sat, mllib_VectorDistance_S8_Sat, mllib_VectorDistance_S16_Sat, mllib_VectorDistance_S32_Sat – vector Euclidean distance

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VectorDistance_U8_Sat(mllib_d64 *z, const mllib_u8
    *x, const mllib_u8 *y, mllib_s32 n);

mllib_status mllib_VectorDistance_S8_Sat(mllib_d64 *z, const mllib_s8
    *x, const mllib_s8 *y, mllib_s32 n);

mllib_status mllib_VectorDistance_S16_Sat(mllib_d64 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 n);

mllib_status mllib_VectorDistance_S32_Sat(mllib_d64 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 n);
```

DESCRIPTION Each of these functions computes the Euclidean distances between two vectors. The following equation is used:

$$z[0] = \left\{ \sum_{i=0}^{n-1} (x[i] - y[i])**2 \right\}**0.5$$

PARAMETERS Each of the functions takes the following arguments:

- z* Pointer to the distance between the two vectors.
- x* Pointer to the first element of the first source vector.
- y* Pointer to the first element of the second source vector.
- n* Number of elements in the vectors.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO attributes(5)

mllib_VectorDotProd_U8_Sat(3MLIB)

NAME	mllib_VectorDotProd_U8_Sat, mllib_VectorDotProd_U8C_Sat, mllib_VectorDotProd_S8_Sat, mllib_VectorDotProd_S8C_Sat, mllib_VectorDotProd_S16_Sat, mllib_VectorDotProd_S16C_Sat, mllib_VectorDotProd_S32_Sat, mllib_VectorDotProd_S32C_Sat – vector dot product (inner product)
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VectorDotProd_U8_Sat(mllib_d64 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 n); mllib_status mllib_VectorDotProd_U8C_Sat(mllib_d64 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 n); mllib_status mllib_VectorDotProd_S8_Sat(mllib_d64 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 n); mllib_status mllib_VectorDotProd_S8C_Sat(mllib_d64 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 n); mllib_status mllib_VectorDotProd_S16_Sat(mllib_d64 *z, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 n); mllib_status mllib_VectorDotProd_S16C_Sat(mllib_d64 *z, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 n); mllib_status mllib_VectorDotProd_S32_Sat(mllib_d64 *z, const mllib_s32 *x, const mllib_s32 *y, mllib_s32 n); mllib_status mllib_VectorDotProd_S32C_Sat(mllib_d64 *z, const mllib_s32 *x, const mllib_s32 *y, mllib_s32 n);</pre>
DESCRIPTION	<p>Each of these functions computes the dot product of two vectors, defined by the following equation:</p> $z = x \cdot Y^*$ <p>where Y^* is the conjugate of the Y vector.</p> <p>For real data, the following equation is used:</p> $z[0] = \sum_{i=0}^{n-1} (x[i] * y[i])$ <p>For complex data, the following equation is used:</p> $z[0] = \sum_{i=0}^{n-1} (x[2*i] * y[2*i] + x[2*i + 1] * y[2*i + 1])$ $z[1] = \sum_{i=0}^{n-1} (x[2*i + 1] * y[2*i] - x[2*i] * y[2*i + 1])$
PARAMETERS	Each of the functions takes the following arguments:

mllib_VectorDotProd_U8_Sat(3MLIB)

z Pointer to the dot product of the two vectors.
x Pointer to the first element of the first source vector.
y Pointer to the first element of the second source vector.
n Number of elements in the vectors.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `attributes(5)`

mllib_VectorMag_U8C(3MLIB)

NAME	mllib_VectorMag_U8C, mllib_VectorMag_S8C, mllib_VectorMag_S16C, mllib_VectorMag_S32C – vector complex magnitude						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VectorMag_U8C(mllib_d64 *m, const mllib_u8 *x, mllib_s32 n); mllib_status mllib_VectorMag_S8C(mllib_d64 *m, const mllib_s8 *x, mllib_s32 n); mllib_status mllib_VectorMag_S16C(mllib_d64 *m, const mllib_s16 *x, mllib_s32 n); mllib_status mllib_VectorMag_S32C(mllib_d64 *m, const mllib_s32 *x, mllib_s32 n);</pre>						
DESCRIPTION	<p>Each of these functions computes the magnitude vector of the complex input vector.</p> <p>The following equation is used:</p> $m[i] = (x[2*i]**2 + x[2*i + 1]**2)**0.5$ <p>where $i = 0, 1, \dots, (n - 1)$.</p>						
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <p><i>m</i> Pointer to the destination magnitude vector.</p> <p><i>x</i> Pointer to the source vector</p> <p><i>n</i> Number of elements in the vector.</p>						
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	attributes(5)						

mllib_VectorMaximumMag_U8C(3MLIB)

NAME mllib_VectorMaximumMag_U8C, mllib_VectorMaximumMag_S8C, mllib_VectorMaximumMag_S16C, mllib_VectorMaximumMag_S32C, mllib_VectorMaximumMag_F32C, mllib_VectorMaximumMag_D64C – find the first element with the maximum magnitude in a vector

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VectorMaximumMag_U8C(mllib_u8 *max, const mllib_u8
    *x, mllib_s32 n);

mllib_status mllib_VectorMaximumMag_S8C(mllib_s8 *max, const mllib_s8
    *x, mllib_s32 n);

mllib_status mllib_VectorMaximumMag_S16C(mllib_s16 *max, const
    mllib_s16 *x, mllib_s32 n);

mllib_status mllib_VectorMaximumMag_S32C(mllib_s32 *max, const
    mllib_s32 *x, mllib_s32 n);

mllib_status mllib_VectorMaximumMag_F32C(mllib_f32 *max, const
    mllib_f32 *x, mllib_s32 n);

mllib_status mllib_VectorMaximumMag_D64C(mllib_d64 *max, const
    mllib_d64 *x, mllib_s32 n);
```

DESCRIPTION Each of these functions finds the first element with the maximum magnitude in a complex vector, then puts the real and imaginary parts of it into max[0] and max[1], respectively.

PARAMETERS Each of the functions takes the following arguments:

max Pointer to the first element with the maximum magnitude.

x Pointer to the first element of the source vector.

n Number of elements in the source vector.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VectorMinimumMag_U8C(3MLIB), mllib_MatrixMaximumMag_U8C(3MLIB), mllib_MatrixMinimumMag_U8C(3MLIB), attributes(5)

mllib_VectorMaximum_U8(3MLIB)

NAME	mllib_VectorMaximum_U8, mllib_VectorMaximum_S8, mllib_VectorMaximum_S16, mllib_VectorMaximum_S32, mllib_VectorMaximum_F32, mllib_VectorMaximum_D64 – find the maximum value in a vector						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VectorMaximum_U8(mllib_u8 *max, const mllib_u8 *x, mllib_s32 n); mllib_status mllib_VectorMaximum_S8(mllib_s8 *max, const mllib_s8 *x, mllib_s32 n); mllib_status mllib_VectorMaximum_S16(mllib_s16 *max, const mllib_s16 *x, mllib_s32 n); mllib_status mllib_VectorMaximum_S32(mllib_s32 *max, const mllib_s32 *x, mllib_s32 n); mllib_status mllib_VectorMaximum_F32(mllib_f32 *max, const mllib_f32 *x, mllib_s32 n); mllib_status mllib_VectorMaximum_D64(mllib_d64 *max, const mllib_d64 *x, mllib_s32 n);</pre>						
DESCRIPTION	Each of these functions finds the maximum value of all elements in a vector. The following equation is used: $\max[0] = \text{MAX}\{ x[i] \mid i = 0, 1, \dots, (n - 1) \}$						
PARAMETERS	Each of the functions takes the following arguments: <i>max</i> Pointer to the maximum value. <i>x</i> Pointer to the first element of the source vector. <i>n</i> Number of elements in the source vector.						
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_VectorMinimum_U8(3MLIB) , mllib_MatrixMaximum_U8(3MLIB) , mllib_MatrixMinimum_U8(3MLIB) , attributes(5)						

mllib_VectorMerge_U8C_U8(3MLIB)

NAME mllib_VectorMerge_U8C_U8, mllib_VectorMerge_S8C_S8, mllib_VectorMerge_S16C_S16, mllib_VectorMerge_S32C_S32 – vector merge

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VectorMerge_U8C_U8(mllib_u8 *z, const mllib_u8 *r,
const mllib_u8 *i, mllib_s32 n);

mllib_status mllib_VectorMerge_S8C_S8(mllib_s8 *z, const mllib_s8 *r,
const mllib_s8 *i, mllib_s32 n);

mllib_status mllib_VectorMerge_S16C_S16(mllib_s16 *z, const mllib_s16
*r, const mllib_s16 *i, mllib_s32 n);

mllib_status mllib_VectorMerge_S32C_S32(mllib_s32 *z, const mllib_s32
*r, const mllib_s32 *i, mllib_s32 n);
```

DESCRIPTION Each of these functions computes the complex vector from two vectors representing the real and imaginary parts.

The following equation is used:

$$\begin{aligned} z[2*k] &= r[k] \\ z[2*k + 1] &= i[k] \end{aligned}$$

where $k = 0, 1, \dots, (n - 1)$.

PARAMETERS Each of the functions takes the following arguments:

z Pointer to the first complex element of the destination vector. *z*[2*k] contains the real part, and *z*[2*k + 1] contains the imaginary part.

r Pointer to the first element of the real part.

i Pointer to the first element of the imaginary part.

n Number of elements in the vectors.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_VectorSplit_U8_U8C\(3MLIB\)](#), [attributes\(5\)](#)

mllib_VectorMinimumMag_U8C(3MLIB)

NAME	mllib_VectorMinimumMag_U8C, mllib_VectorMinimumMag_S8C, mllib_VectorMinimumMag_S16C, mllib_VectorMinimumMag_S32C, mllib_VectorMinimumMag_F32C, mllib_VectorMinimumMag_D64C – find the first element with the minimum magnitude in a vector						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VectorMinimumMag_U8C(mllib_u8 *min, const mllib_u8 *x, mllib_s32 n); mllib_status mllib_VectorMinimumMag_S8C(mllib_s8 *min, const mllib_s8 *x, mllib_s32 n); mllib_status mllib_VectorMinimumMag_S16C(mllib_s16 *min, const mllib_s16 *x, mllib_s32 n); mllib_status mllib_VectorMinimumMag_S32C(mllib_s32 *min, const mllib_s32 *x, mllib_s32 n); mllib_status mllib_VectorMinimumMag_F32C(mllib_f32 *min, const mllib_f32 *x, mllib_s32 n); mllib_status mllib_VectorMinimumMag_D64C(mllib_d64 *min, const mllib_d64 *x, mllib_s32 n);</pre>						
DESCRIPTION	Each of these functions finds the first element with the minimum magnitude in a complex vector, then puts the real and imaginary parts of it into <code>min[0]</code> and <code>min[1]</code> , respectively.						
PARAMETERS	Each of the functions takes the following arguments: <i>min</i> Pointer to the first element with the minimum magnitude. <i>x</i> Pointer to the first element of the source vector. <i>n</i> Number of elements in the source vector.						
RETURN VALUES	Each of the functions returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1" data-bbox="444 1465 1414 1600"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_VectorMaximumMag_U8C(3MLIB)</code> , <code>mllib_MatrixMaximumMag_U8C(3MLIB)</code> , <code>mllib_MatrixMinimumMag_U8C(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_VectorMinimum_U8, mllib_VectorMinimum_S8, mllib_VectorMinimum_S16, mllib_VectorMinimum_S32, mllib_VectorMinimum_F32, mllib_VectorMinimum_D64 – find the minimum value in a vector

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VectorMinimum_U8(mllib_u8 *min, const mllib_u8 *x,
    mllib_s32 n);

mllib_status mllib_VectorMinimum_S8(mllib_s8 *min, const mllib_s8 *x,
    mllib_s32 n);

mllib_status mllib_VectorMinimum_S16(mllib_s16 *min, const mllib_s16
    *x, mllib_s32 n);

mllib_status mllib_VectorMinimum_S32(mllib_s32 *min, const mllib_s32
    *x, mllib_s32 n);

mllib_status mllib_VectorMinimum_F32(mllib_f32 *min, const mllib_f32
    *x, mllib_s32 n);

mllib_status mllib_VectorMinimum_D64(mllib_d64 *min, const mllib_d64
    *x, mllib_s32 n);
```

DESCRIPTION Each of these functions finds the minimum value of all elements in a vector.

The following equation is used:

$$\max[0] = \text{MIN}\{ x[i] \quad i = 0, 1, \dots, (n - 1) \}$$

PARAMETERS Each of the functions takes the following arguments:

- min* Pointer to the minimum value.
- x* Pointer to the first element of the source vector.
- n* Number of elements in the source vector.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_VectorMaximum_U8\(3MLIB\)](#), [mllib_MatrixMaximum_U8\(3MLIB\)](#), [mllib_MatrixMinimum_U8\(3MLIB\)](#), [attributes\(5\)](#)

mllib_VectorMulMShift_S16_S16_Mod(3MLIB)

NAME	mllib_VectorMulMShift_S16_S16_Mod, mllib_VectorMulMShift_S16_S16_Sat, mllib_VectorMulMShift_S16C_S16C_Mod, mllib_VectorMulMShift_S16C_S16C_Sat – multiplication of vector by matrix with shifting
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VectorMulMShift_S16_S16_Mod(mllib_s16 *z, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulMShift_S16_S16_Sat(mllib_s16 *z, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulMShift_S16C_S16C_Mod(mllib_s16 *z, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulMShift_S16C_S16C_Sat(mllib_s16 *z, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n, mllib_s32 shift);</pre>
DESCRIPTION	<p>Each of these functions multiplies a vector by a matrix and shifts the results.</p> <p>For real data, the following equation is used:</p> $z[i] = \left\{ \sum_{j=0}^{m-1} (x[j] * y[j*m + i]) \right\} * 2^{**(-shift)}$ <p>where $i = 0, 1, \dots, (n - 1)$.</p> <p>For complex data, the following equation is used:</p> $z[2*i] = \left\{ \sum_{j=0}^{m-1} (xR*yR - xI*yI) \right\} * 2^{**(-shift)}$ $z[2*i + 1] = \left\{ \sum_{j=0}^{m-1} (xR*yI + xI*yR) \right\} * 2^{**(-shift)}$ <p>where $i = 0, 1, \dots, (n - 1)$, and</p> <pre>xR = x[2*j] xI = x[2*j + 1] yR = y[2*(j*m + i)] yI = y[2*(j*m + i) + 1]</pre>
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <p>z Pointer to the first element of the destination vector.</p> <p>x Pointer to the first element of the source vector.</p> <p>y Pointer to the first element of the source matrix.</p>

`mllib_VectorMulMShift_S16_S16_Mod(3MLIB)`

m Number of rows in the matrix, and number of elements in the source vector.

n Number of columns in the matrix, and number of elements in the destination vector.

shift Right shifting factor.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_VectorMulM_U8_U8_Mod(3MLIB)`, `attributes(5)`

mllib_VectorMulM_U8_U8_Mod(3MLIB)

NAME	mllib_VectorMulM_U8_U8_Mod, mllib_VectorMulM_U8_U8_Sat, mllib_VectorMulM_U8C_U8C_Mod, mllib_VectorMulM_U8C_U8C_Sat, mllib_VectorMulM_S8_S8_Mod, mllib_VectorMulM_S8_S8_Sat, mllib_VectorMulM_S8C_S8C_Mod, mllib_VectorMulM_S8C_S8C_Sat, mllib_VectorMulM_S16_U8_Mod, mllib_VectorMulM_S16_U8_Sat, mllib_VectorMulM_S16_S8_Mod, mllib_VectorMulM_S16_S8_Sat, mllib_VectorMulM_S16_S16_Mod, mllib_VectorMulM_S16_S16_Sat, mllib_VectorMulM_S16C_U8C_Mod, mllib_VectorMulM_S16C_U8C_Sat, mllib_VectorMulM_S16C_S8C_Mod, mllib_VectorMulM_S16C_S8C_Sat, mllib_VectorMulM_S16C_S16C_Mod, mllib_VectorMulM_S16C_S16C_Sat, mllib_VectorMulM_S32_S16_Mod, mllib_VectorMulM_S32_S16_Sat, mllib_VectorMulM_S32_S32_Mod, mllib_VectorMulM_S32_S32_Sat, mllib_VectorMulM_S32C_S16C_Mod, mllib_VectorMulM_S32C_S16C_Sat, mllib_VectorMulM_S32C_S32C_Mod, mllib_VectorMulM_S32C_S32C_Sat – multiplication of vector by matrix
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VectorMulM_U8_U8_Mod(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_VectorMulM_U8_U8_Sat(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_VectorMulM_U8C_U8C_Mod(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_VectorMulM_U8C_U8C_Sat(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_VectorMulM_S8_S8_Mod(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_VectorMulM_S8_S8_Sat(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_VectorMulM_S8C_S8C_Mod(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_VectorMulM_S8C_S8C_Sat(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_VectorMulM_S16_U8_Mod(mllib_s16 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_VectorMulM_S16_U8_Sat(mllib_s16 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_VectorMulM_S16_S8_Mod(mllib_s16 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 n); mllib_status mllib_VectorMulM_S16_S8_Sat(mllib_s16 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 n);</pre>

mllib_VectorMulM_U8_U8_Mod(3MLIB)

```

mllib_status mllib_VectorMulM_S16_S16_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_VectorMulM_S16_S16_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_VectorMulM_S16C_U8C_Mod(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_VectorMulM_S16C_U8C_Sat(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_VectorMulM_S16C_S8C_Mod(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_VectorMulM_S16C_S8C_Sat(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_VectorMulM_S16C_S16C_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_VectorMulM_S16C_S16C_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_VectorMulM_S32_S16_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_VectorMulM_S32_S16_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_VectorMulM_S32_S32_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_VectorMulM_S32_S32_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_VectorMulM_S32C_S16C_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_VectorMulM_S32C_S16C_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_VectorMulM_S32C_S32C_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 m, mllib_s32 n);
mllib_status mllib_VectorMulM_S32C_S32C_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 m, mllib_s32 n);

```

DESCRIPTION

Each of these functions multiplies a vector by a matrix.

For real data, the following equation is used:

$$z[i] = \sum_{j=0}^{m-1} (x[j] * y[j*m + i])$$

where $i = 0, 1, \dots, (n - 1)$.

mllib_VectorMulM_U8_U8_Mod(3MLIB)

For complex data, the following equation is used:

$$z[2*i] = \sum_{j=0}^{m-1} (xR*yR - xI*yI)$$

$$z[2*i + 1] = \sum_{j=0}^{m-1} (xR*yI + xI*yR)$$

where $i = 0, 1, \dots, (n - 1)$, and

$xR = x[2*j]$
 $xI = x[2*j + 1]$
 $yR = y[2*(j*m + i)]$
 $yI = y[2*(j*m + i) + 1]$

PARAMETERS Each of the functions takes the following arguments:

z Pointer to the first element of the destination vector.
x Pointer to the first element of the source vector.
y Pointer to the first element of the source matrix.
m Number of rows in the matrix, and number of elements in the source vector.
n Number of columns in the matrix, and number of elements in the destination vector.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_VectorMulMShift_S16_S16_Mod\(3MLIB\)](#), [attributes\(5\)](#)

mllib_VectorMulSAdd_U8_Mod(3MLIB)

NAME mllib_VectorMulSAdd_U8_Mod, mllib_VectorMulSAdd_U8_Sat, mllib_VectorMulSAdd_U8C_Mod, mllib_VectorMulSAdd_U8C_Sat, mllib_VectorMulSAdd_S8_Mod, mllib_VectorMulSAdd_S8_Sat, mllib_VectorMulSAdd_S8C_Mod, mllib_VectorMulSAdd_S8C_Sat, mllib_VectorMulSAdd_S16_Mod, mllib_VectorMulSAdd_S16_Sat, mllib_VectorMulSAdd_S16C_Mod, mllib_VectorMulSAdd_S16C_Sat, mllib_VectorMulSAdd_S32_Mod, mllib_VectorMulSAdd_S32_Sat, mllib_VectorMulSAdd_S32C_Mod, mllib_VectorMulSAdd_S32C_Sat – vector multiplication by scalar plus addition, in place

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VectorMulSAdd_U8_Mod(mllib_u8 *xz, const mllib_u8
    *y, const mllib_u8 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_U8_Sat(mllib_u8 *xz, const mllib_u8
    *y, const mllib_u8 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_U8C_Mod(mllib_u8 *xz, const mllib_u8
    *y, const mllib_u8 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_U8C_Sat(mllib_u8 *xz, const mllib_u8
    *y, const mllib_u8 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_S8_Mod(mllib_s8 *xz, const mllib_s8
    *y, const mllib_s8 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_S8_Sat(mllib_s8 *xz, const mllib_s8
    *y, const mllib_s8 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_S8C_Mod(mllib_s8 *xz, const mllib_s8
    *y, const mllib_s8 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_S8C_Sat(mllib_s8 *xz, const mllib_s8
    *y, const mllib_s8 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_S16_Mod(mllib_s16 *xz, const
    mllib_s16 *y, const mllib_s16 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_S16_Sat(mllib_s16 *xz, const
    mllib_s16 *y, const mllib_s16 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_S16C_Mod(mllib_s16 *xz, const
    mllib_s16 *y, const mllib_s16 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_S16C_Sat(mllib_s16 *xz, const
    mllib_s16 *y, const mllib_s16 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_S32_Mod(mllib_s32 *xz, const
    mllib_s32 *y, const mllib_s32 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_S32_Sat(mllib_s32 *xz, const
    mllib_s32 *y, const mllib_s32 *c, mllib_s32 n);
```

mllib_VectorMulSAdd_U8_Mod(3MLIB)

```
mllib_status mllib_VectorMulSAdd_S32C_Mod(mllib_s32 *xz, const
mllib_s32 *y, const mllib_s32 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_S32C_Sat(mllib_s32 *xz, const
mllib_s32 *y, const mllib_s32 *c, mllib_s32 n);
```

DESCRIPTION Each of these functions computes an in-place multiplication of a vector by a scalar and adds the result to another vector.

For real data, the following equation is used:

$$xz[i] = xz[i] + y[i]*c[0]$$

where $i = 0, 1, \dots, (n - 1)$.

For complex data, the following equation is used:

$$\begin{aligned} xz[2*i] &= xz[2*i] + y[2*i]*c[0] - y[2*i + 1]*c[1] \\ xz[2*i + 1] &= xz[2*i + 1] + y[2*i]*c[1] + y[2*i + 1]*c[0] \end{aligned}$$

where $i = 0, 1, \dots, (n - 1)$.

PARAMETERS Each of the functions takes the following arguments:

xz Pointer to the first element of the first source and destination vector.

y Pointer to the first element of the second source vector.

c Pointer to the source scalar. When the function is used with complex data types, *c*[0] contains the real part of the scalar, and *c*[1] contains the imaginary part of the scalar.

n Number of elements in the vectors.

RETURN VALUES Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_VectorMulSAdd_U8_U8_Mod\(3MLIB\)](#), [attributes\(5\)](#)

mllib_VectorMulSAdd_U8_U8_Mod(3MLIB)

NAME mllib_VectorMulSAdd_U8_U8_Mod, mllib_VectorMulSAdd_U8_U8_Sat, mllib_VectorMulSAdd_U8C_U8C_Mod, mllib_VectorMulSAdd_U8C_U8C_Sat, mllib_VectorMulSAdd_S8_S8_Mod, mllib_VectorMulSAdd_S8_S8_Sat, mllib_VectorMulSAdd_S8C_S8C_Mod, mllib_VectorMulSAdd_S8C_S8C_Sat, mllib_VectorMulSAdd_S16_U8_Mod, mllib_VectorMulSAdd_S16_U8_Sat, mllib_VectorMulSAdd_S16_S8_Mod, mllib_VectorMulSAdd_S16_S8_Sat, mllib_VectorMulSAdd_S16_S16_Mod, mllib_VectorMulSAdd_S16_S16_Sat, mllib_VectorMulSAdd_S16C_U8C_Mod, mllib_VectorMulSAdd_S16C_U8C_Sat, mllib_VectorMulSAdd_S16C_S8C_Mod, mllib_VectorMulSAdd_S16C_S8C_Sat, mllib_VectorMulSAdd_S16C_S16C_Mod, mllib_VectorMulSAdd_S16C_S16C_Sat, mllib_VectorMulSAdd_S32_S16_Mod, mllib_VectorMulSAdd_S32_S16_Sat, mllib_VectorMulSAdd_S32_S32_Mod, mllib_VectorMulSAdd_S32_S32_Sat, mllib_VectorMulSAdd_S32C_S16C_Mod, mllib_VectorMulSAdd_S32C_S16C_Sat, mllib_VectorMulSAdd_S32C_S32C_Mod, mllib_VectorMulSAdd_S32C_S32C_Sat – vector multiplication by scalar plus addition

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VectorMulSAdd_U8_U8_Mod(mllib_u8 *z, const
    mllib_u8 *x, const mllib_u8 *y, const mllib_u8 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_U8_U8_Sat(mllib_u8 *z, const
    mllib_u8 *x, const mllib_u8 *y, const mllib_u8 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_U8C_U8C_Mod(mllib_u8 *z, const
    mllib_u8 *x, const mllib_u8 *y, const mllib_u8 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_U8C_U8C_Sat(mllib_u8 *z, const
    mllib_u8 *x, const mllib_u8 *y, const mllib_u8 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_S8_S8_Mod(mllib_s8 *z, const
    mllib_s8 *x, const mllib_s8 *y, const mllib_s8 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_S8_S8_Sat(mllib_s8 *z, const
    mllib_s8 *x, const mllib_s8 *y, const mllib_s8 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_S8C_S8C_Mod(mllib_s8 *z, const
    mllib_s8 *x, const mllib_s8 *y, const mllib_s8 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_S8C_S8C_Sat(mllib_s8 *z, const
    mllib_s8 *x, const mllib_s8 *y, const mllib_s8 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_S16_U8_Mod(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *y, const mllib_u8 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_S16_U8_Sat(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *y, const mllib_u8 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_S16_S8_Mod(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *y, const mllib_s8 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_S16_S8_Sat(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *y, const mllib_s8 *c, mllib_s32 n);
```

mllib_VectorMulSAdd_U8_U8_Mod(3MLIB)

```
mllib_status mllib_VectorMulSAdd_S16_S16_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *c, mllib_s32
    n);

mllib_status mllib_VectorMulSAdd_S16_S16_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *c, mllib_s32
    n);

mllib_status mllib_VectorMulSAdd_S16C_U8C_Mod(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *y, const mllib_u8 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_S16C_U8C_Sat(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *y, const mllib_u8 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_S16C_S8C_Mod(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *y, const mllib_s8 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_S16C_S8C_Sat(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *y, const mllib_s8 *c, mllib_s32 n);

mllib_status mllib_VectorMulSAdd_S16C_S16C_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *c, mllib_s32
    n);

mllib_status mllib_VectorMulSAdd_S16C_S16C_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *c, mllib_s32
    n);

mllib_status mllib_VectorMulSAdd_S32_S16_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *c, mllib_s32
    n);

mllib_status mllib_VectorMulSAdd_S32_S16_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *c, mllib_s32
    n);

mllib_status mllib_VectorMulSAdd_S32_S32_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, const mllib_s32 *c, mllib_s32
    n);

mllib_status mllib_VectorMulSAdd_S32_S32_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, const mllib_s32 *c, mllib_s32
    n);

mllib_status mllib_VectorMulSAdd_S32C_S16C_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *c, mllib_s32
    n);

mllib_status mllib_VectorMulSAdd_S32C_S16C_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, const mllib_s16 *c, mllib_s32
    n);

mllib_status mllib_VectorMulSAdd_S32C_S32C_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, const mllib_s32 *c, mllib_s32
    n);
```


mlib_VectorMulSAdd_U8_U8_Mod(3MLIB)

```
mlib_status mlib_VectorMulSAdd_S32C_S32C_Sat(mlib_s32 *z, const
mlib_s32 *x, const mlib_s32 *y, const mlib_s32 *c, mlib_s32
n);
```

DESCRIPTION Each of these functions multiplies a vector by a scalar and adds the result to another vector.

For real data, the following equation is used:

$$z[i] = x[i] + y[i]*c[0]$$

where $i = 0, 1, \dots, (n - 1)$.

For complex data, the following equation is used:

$$\begin{aligned} z[2*i] &= x[2*i] + y[2*i]*c[0] - y[2*i + 1]*c[1] \\ z[2*i + 1] &= x[2*i + 1] + y[2*i]*c[1] + y[2*i + 1]*c[0] \end{aligned}$$

where $i = 0, 1, \dots, (n - 1)$.

PARAMETERS Each of the functions takes the following arguments:

- z* Pointer to the first element of the destination vector.
- x* Pointer to the first element of the first source vector.
- y* Pointer to the first element of the second source vector.
- c* Pointer to the source scalar. When the function is used with complex data types, *c*[0] contains the real part of the scalar, and *c*[1] contains the imaginary part of the scalar.
- n* Number of elements in the vectors.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mlib_VectorMulSAdd_U8_Mod(3MLIB)`, `attributes(5)`

mllib_VectorMulShift_U8_Mod(3MLIB)

NAME	mllib_VectorMulShift_U8_Mod, mllib_VectorMulShift_U8_Sat, mllib_VectorMulShift_U8C_Mod, mllib_VectorMulShift_U8C_Sat, mllib_VectorMulShift_S8_Mod, mllib_VectorMulShift_S8_Sat, mllib_VectorMulShift_S8C_Mod, mllib_VectorMulShift_S8C_Sat, mllib_VectorMulShift_S16_Mod, mllib_VectorMulShift_S16_Sat, mllib_VectorMulShift_S16C_Mod, mllib_VectorMulShift_S16C_Sat, mllib_VectorMulShift_S32_Mod, mllib_VectorMulShift_S32_Sat, mllib_VectorMulShift_S32C_Mod, mllib_VectorMulShift_S32C_Sat – vector multiplication with shifting, in place
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VectorMulShift_U8_Mod(mllib_u8 *xz, const mllib_u8 *y, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulShift_U8_Sat(mllib_u8 *xz, const mllib_u8 *y, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulShift_U8C_Mod(mllib_u8 *xz, const mllib_u8 *y, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulShift_U8C_Sat(mllib_u8 *xz, const mllib_u8 *y, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulShift_S8_Mod(mllib_s8 *xz, const mllib_s8 *y, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulShift_S8_Sat(mllib_s8 *xz, const mllib_s8 *y, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulShift_S8C_Mod(mllib_s8 *xz, const mllib_s8 *y, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulShift_S8C_Sat(mllib_s8 *xz, const mllib_s8 *y, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulShift_S16_Mod(mllib_s16 *xz, const mllib_s16 *y, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulShift_S16_Sat(mllib_s16 *xz, const mllib_s16 *y, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulShift_S16C_Mod(mllib_s16 *xz, const mllib_s16 *y, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulShift_S16C_Sat(mllib_s16 *xz, const mllib_s16 *y, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulShift_S32_Mod(mllib_s32 *xz, const mllib_s32 *y, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulShift_S32_Sat(mllib_s32 *xz, const mllib_s32 *y, mllib_s32 n, mllib_s32 shift);</pre>

mllib_VectorMulShift_U8_Mod(3MLIB)

```
mllib_status mllib_VectorMulShift_S32C_Mod(mllib_s32 *xz, const
mllib_s32 *y, mllib_s32 n, mllib_s32 shift);

mllib_status mllib_VectorMulShift_S32C_Sat(mllib_s32 *xz, const
mllib_s32 *y, mllib_s32 n, mllib_s32 shift);
```

DESCRIPTION Each of these functions performs an in-place multiplication of two vectors and shifts the result.

For real data, the following equation is used:

$$xz[i] = xz[i] * y[i] * 2^{**(-shift)}$$

where $i = 0, 1, \dots, (n - 1)$.

For complex data, the following equation is used:

$$\begin{aligned} \text{tmp} &= xz[2*i] \\ xz[2*i] &= (\text{tmp}*y[2*i] - xz[2*i + 1]*y[2*i + 1]) * 2^{**(-shift)} \\ xz[2*i + 1] &= (\text{tmp}*y[2*i + 1] + xz[2*i + 1]*y[2*i]) * 2^{**(-shift)} \end{aligned}$$

where $i = 0, 1, \dots, (n - 1)$.

PARAMETERS Each of the functions takes the following arguments:

xz Pointer to the first element of the first source and result vector.

y Pointer to the first element of the second source vector.

n Number of elements in each vector.

shift Right shifting factor. The ranges of valid *shift* are:

$1 \leq \text{shift} \leq 8$ for U8, S8, U8C, S8C types

$1 \leq \text{shift} \leq 16$ for S16, S16C types

$1 \leq \text{shift} \leq 31$ for S32, S32C types

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_VectorMulShift_U8_U8_Mod(3MLIB)`, `attributes(5)`

mllib_VectorMulShift_U8_U8_Mod(3MLIB)

NAME	mllib_VectorMulShift_U8_U8_Mod, mllib_VectorMulShift_U8_U8_Sat, mllib_VectorMulShift_U8C_U8C_Mod, mllib_VectorMulShift_U8C_U8C_Sat, mllib_VectorMulShift_S8_S8_Mod, mllib_VectorMulShift_S8_S8_Sat, mllib_VectorMulShift_S8C_S8C_Mod, mllib_VectorMulShift_S8C_S8C_Sat, mllib_VectorMulShift_S16_S16_Mod, mllib_VectorMulShift_S16_S16_Sat, mllib_VectorMulShift_S16C_S16C_Mod, mllib_VectorMulShift_S16C_S16C_Sat, mllib_VectorMulShift_S32_S32_Mod, mllib_VectorMulShift_S32_S32_Sat, mllib_VectorMulShift_S32C_S32C_Mod, mllib_VectorMulShift_S32C_S32C_Sat – vector multiplication with shifting
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VectorMulShift_U8_U8_Mod(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulShift_U8_U8_Sat(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulShift_U8C_U8C_Mod(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulShift_U8C_U8C_Sat(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulShift_S8_S8_Mod(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulShift_S8_S8_Sat(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulShift_S8C_S8C_Mod(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulShift_S8C_S8C_Sat(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulShift_S16_S16_Mod(mllib_s16 *z, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulShift_S16_S16_Sat(mllib_s16 *z, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulShift_S16C_S16C_Mod(mllib_s16 *z, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulShift_S16C_S16C_Sat(mllib_s16 *z, const mllib_s16 *x, const mllib_s16 *y, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulShift_S32_S32_Mod(mllib_s32 *z, const mllib_s32 *x, const mllib_s32 *y, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulShift_S32_S32_Sat(mllib_s32 *z, const mllib_s32 *x, const mllib_s32 *y, mllib_s32 n, mllib_s32 shift);</pre>

mllib_VectorMulShift_U8_U8_Mod(3MLIB)

```
mllib_status mllib_VectorMulShift_S32C_S32C_Mod(mllib_s32 *z, const
mllib_s32 *x, const mllib_s32 *y, mllib_s32 n, mllib_s32 shift);

mllib_status mllib_VectorMulShift_S32C_S32C_Sat(mllib_s32 *z, const
mllib_s32 *x, const mllib_s32 *y, mllib_s32 n, mllib_s32 shift);
```

DESCRIPTION

Each of these functions performs a multiplication of two vectors, shifts the result, and puts it into a third vector.

For real data, the following equation is used:

$$z[i] = x[i] * y[i] * 2^{**(-shift)}$$

where $i = 0, 1, \dots, (n - 1)$.

For complex data, the following equation is used:

$$\begin{aligned} z[2*i] &= (x[2*i]*y[2*i] - x[2*i + 1]*y[2*i + 1]) * 2^{**(-shift)} \\ z[2*i + 1] &= (x[2*i]*y[2*i + 1] + x[2*i + 1]*y[2*i]) * 2^{**(-shift)} \end{aligned}$$

where $i = 0, 1, \dots, (n - 1)$.

PARAMETERS

Each of the functions takes the following arguments:

- z* Pointer to the first element of the result vector.
- x* Pointer to the first element of the first source vector.
- y* Pointer to the first element of the second source vector.
- n* Number of elements in each vector.
- shift* Right shifting factor. The ranges of valid *shift* are:
 - 1 ≤ *shift* ≤ 8 for U8, S8, U8C, S8C types
 - 1 ≤ *shift* ≤ 16 for S16, S16C types
 - 1 ≤ *shift* ≤ 31 for S32, S32C types

RETURN VALUES

Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

`mllib_VectorMulShift_U8_Mod(3MLIB)`, `attributes(5)`

mllib_VectorMulSShift_U8_Mod(3MLIB)

NAME	mllib_VectorMulSShift_U8_Mod, mllib_VectorMulSShift_U8_Sat, mllib_VectorMulSShift_U8C_Mod, mllib_VectorMulSShift_U8C_Sat, mllib_VectorMulSShift_S8_Mod, mllib_VectorMulSShift_S8_Sat, mllib_VectorMulSShift_S8C_Mod, mllib_VectorMulSShift_S8C_Sat, mllib_VectorMulSShift_S16_Mod, mllib_VectorMulSShift_S16_Sat, mllib_VectorMulSShift_S16C_Mod, mllib_VectorMulSShift_S16C_Sat, mllib_VectorMulSShift_S32_Mod, mllib_VectorMulSShift_S32_Sat, mllib_VectorMulSShift_S32C_Mod, mllib_VectorMulSShift_S32C_Sat – vector multiplication by scalar plus shifting, in place
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VectorMulSShift_U8_Mod(mllib_u8 *xz, const mllib_u8 *c, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulSShift_U8_Sat(mllib_u8 *xz, const mllib_u8 *c, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulSShift_U8C_Mod(mllib_u8 *xz, const mllib_u8 *c, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulSShift_U8C_Sat(mllib_u8 *xz, const mllib_u8 *c, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulSShift_S8_Mod(mllib_s8 *xz, const mllib_s8 *c, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulSShift_S8_Sat(mllib_s8 *xz, const mllib_s8 *c, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulSShift_S8C_Mod(mllib_s8 *xz, const mllib_s8 *c, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulSShift_S8C_Sat(mllib_s8 *xz, const mllib_s8 *c, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulSShift_S16_Mod(mllib_s16 *xz, const mllib_s16 *c, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulSShift_S16_Sat(mllib_s16 *xz, const mllib_s16 *c, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulSShift_S16C_Mod(mllib_s16 *xz, const mllib_s16 *c, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulSShift_S16C_Sat(mllib_s16 *xz, const mllib_s16 *c, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulSShift_S32_Mod(mllib_s32 *xz, const mllib_s32 *c, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulSShift_S32_Sat(mllib_s32 *xz, const mllib_s32 *c, mllib_s32 n, mllib_s32 shift);</pre>

mllib_VectorMulSShift_U8_Mod(3MLIB)

```
mllib_status mllib_VectorMulSShift_S32C_Mod(mllib_s32 *xz, const
      mllib_s32 *c, mllib_s32 n, mllib_s32 shift);
mllib_status mllib_VectorMulSShift_S32C_Sat(mllib_s32 *xz, const
      mllib_s32 *c, mllib_s32 n, mllib_s32 shift);
```

DESCRIPTION

Each of these functions performs an in-place multiplication of a vector by a scalar and shifts the result.

For real data, the following equation is used:

$$xz[i] = xz[i] * c[0] * 2^{**(-shift)}$$

where $i = 0, 1, \dots, (n - 1)$.

For complex data, the following equation is used:

```
tmp          = xz[2*i]
xz[2*i]      = (tmp*c[0] - xz[2*i + 1]*c[1]) * 2^{**(-shift)}
xz[2*i + 1] = (tmp*c[1] + xz[2*i + 1]*c[0]) * 2^{**(-shift)}
```

where $i = 0, 1, \dots, (n - 1)$.

PARAMETERS

Each of the functions takes the following arguments:

- xz* Pointer to the first element of the source and result vector.
- c* Pointer to the source scalar. When the function is used with complex data types, *c*[0] contains the real part of the scalar, and *c*[1] contains the imaginary part of the scalar.
- n* Number of elements in each vector.
- shift* Right shifting factor. The ranges of valid *shift* are:
 - $1 \leq shift \leq 8$ for U8, S8, U8C, S8C types
 - $1 \leq shift \leq 16$ for S16, S16C types
 - $1 \leq shift \leq 31$ for S32, S32C types

RETURN VALUES

Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES

See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

[mllib_VectorMulSShift_U8_U8_Mod\(3MLIB\)](#), [attributes\(5\)](#)

mllib_VectorMulSShift_U8_U8_Mod(3MLIB)

NAME	mllib_VectorMulSShift_U8_U8_Mod, mllib_VectorMulSShift_U8_U8_Sat, mllib_VectorMulSShift_U8C_U8C_Mod, mllib_VectorMulSShift_U8C_U8C_Sat, mllib_VectorMulSShift_S8_S8_Mod, mllib_VectorMulSShift_S8_S8_Sat, mllib_VectorMulSShift_S8C_S8C_Mod, mllib_VectorMulSShift_S8C_S8C_Sat, mllib_VectorMulSShift_S16_S16_Mod, mllib_VectorMulSShift_S16_S16_Sat, mllib_VectorMulSShift_S16C_S16C_Mod, mllib_VectorMulSShift_S16C_S16C_Sat, mllib_VectorMulSShift_S32_S32_Mod, mllib_VectorMulSShift_S32_S32_Sat, mllib_VectorMulSShift_S32C_S32C_Mod, mllib_VectorMulSShift_S32C_S32C_Sat – vector multiplication by scalar plus shifting
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VectorMulSShift_U8_U8_Mod(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *c, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulSShift_U8_U8_Sat(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *c, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulSShift_U8C_U8C_Mod(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *c, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulSShift_U8C_U8C_Sat(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *c, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulSShift_S8_S8_Mod(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *c, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulSShift_S8_S8_Sat(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *c, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulSShift_S8C_S8C_Mod(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *c, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulSShift_S8C_S8C_Sat(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *c, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulSShift_S16_S16_Mod(mllib_s16 *z, const mllib_s16 *x, const mllib_s16 *c, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulSShift_S16_S16_Sat(mllib_s16 *z, const mllib_s16 *x, const mllib_s16 *c, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulSShift_S16C_S16C_Mod(mllib_s16 *z, const mllib_s16 *x, const mllib_s16 *c, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulSShift_S16C_S16C_Sat(mllib_s16 *z, const mllib_s16 *x, const mllib_s16 *c, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulSShift_S32_S32_Mod(mllib_s32 *z, const mllib_s32 *x, const mllib_s32 *c, mllib_s32 n, mllib_s32 shift); mllib_status mllib_VectorMulSShift_S32_S32_Sat(mllib_s32 *z, const mllib_s32 *x, const mllib_s32 *c, mllib_s32 n, mllib_s32 shift);</pre>

mllib_VectorMulSShift_U8_U8_Mod(3MLIB)

```
mllib_status mllib_VectorMulSShift_S32C_S32C_Mod(mllib_s32 *z, const
mllib_s32 *x, const mllib_s32 *c, mllib_s32 n, mllib_s32 shift);

mllib_status mllib_VectorMulSShift_S32C_S32C_Sat(mllib_s32 *z, const
mllib_s32 *x, const mllib_s32 *c, mllib_s32 n, mllib_s32 shift);
```

DESCRIPTION

Each of these functions performs a multiplication of a vector by a scalar and shifts the result.

For real data, the following equation is used:

$$z[i] = x[i] * c[0] * 2^{**(-shift)}$$

where $i = 0, 1, \dots, (n - 1)$.

For complex data, the following equation is used:

$$\begin{aligned} z[2*i] &= (x[2*i]*c[0] - x[2*i + 1]*c[1]) * 2^{**(-shift)} \\ z[2*i + 1] &= (x[2*i]*c[1] + x[2*i + 1]*c[0]) * 2^{**(-shift)} \end{aligned}$$

where $i = 0, 1, \dots, (n - 1)$.

PARAMETERS

Each of the functions takes the following arguments:

- z* Pointer to the first element of the result vector.
- x* Pointer to the first element of the source vector.
- c* Pointer to the source scalar. When the function is used with complex data types, *c*[0] contains the real part of the scalar, and *c*[1] contains the imaginary part of the scalar.
- n* Number of elements in each vector.
- shift* Right shifting factor. The ranges of valid *shift* are:
 - 1 ≤ *shift* ≤ 8 for U8, S8, U8C, S8C types
 - 1 ≤ *shift* ≤ 16 for S16, S16C types
 - 1 ≤ *shift* ≤ 31 for S32, S32C types

RETURN VALUES

Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES

See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

`mllib_VectorMulSShift_U8_Mod(3MLIB)`, `attributes(5)`

mllib_VectorMulS_U8_Mod(3MLIB)

NAME	mllib_VectorMulS_U8_Mod, mllib_VectorMulS_U8_Sat, mllib_VectorMulS_U8C_Mod, mllib_VectorMulS_U8C_Sat, mllib_VectorMulS_S8_Mod, mllib_VectorMulS_S8_Sat, mllib_VectorMulS_S8C_Mod, mllib_VectorMulS_S8C_Sat, mllib_VectorMulS_S16_Mod, mllib_VectorMulS_S16_Sat, mllib_VectorMulS_S16C_Mod, mllib_VectorMulS_S16C_Sat, mllib_VectorMulS_S32_Mod, mllib_VectorMulS_S32_Sat, mllib_VectorMulS_S32C_Mod, mllib_VectorMulS_S32C_Sat – vector multiplication by scalar, in place
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VectorMulS_U8_Mod(mllib_u8 *xz, const mllib_u8 *c, mllib_s32 n); mllib_status mllib_VectorMulS_U8_Sat(mllib_u8 *xz, const mllib_u8 *c, mllib_s32 n); mllib_status mllib_VectorMulS_U8C_Mod(mllib_u8 *xz, const mllib_u8 *c, mllib_s32 n); mllib_status mllib_VectorMulS_U8C_Sat(mllib_u8 *xz, const mllib_u8 *c, mllib_s32 n); mllib_status mllib_VectorMulS_S8_Mod(mllib_s8 *xz, const mllib_s8 *c, mllib_s32 n); mllib_status mllib_VectorMulS_S8_Sat(mllib_s8 *xz, const mllib_s8 *c, mllib_s32 n); mllib_status mllib_VectorMulS_S8C_Mod(mllib_s8 *xz, const mllib_s8 *c, mllib_s32 n); mllib_status mllib_VectorMulS_S8C_Sat(mllib_s8 *xz, const mllib_s8 *c, mllib_s32 n); mllib_status mllib_VectorMulS_S16_Mod(mllib_s16 *xz, const mllib_s16 *c, mllib_s32 n); mllib_status mllib_VectorMulS_S16_Sat(mllib_s16 *xz, const mllib_s16 *c, mllib_s32 n); mllib_status mllib_VectorMulS_S16C_Mod(mllib_s16 *xz, const mllib_s16 *c, mllib_s32 n); mllib_status mllib_VectorMulS_S16C_Sat(mllib_s16 *xz, const mllib_s16 *c, mllib_s32 n); mllib_status mllib_VectorMulS_S32_Mod(mllib_s32 *xz, const mllib_s32 *c, mllib_s32 n); mllib_status mllib_VectorMulS_S32_Sat(mllib_s32 *xz, const mllib_s32 *c, mllib_s32 n); mllib_status mllib_VectorMulS_S32C_Mod(mllib_s32 *xz, const mllib_s32 *c, mllib_s32 n); mllib_status mllib_VectorMulS_S32C_Sat(mllib_s32 *xz, const mllib_s32 *c, mllib_s32 n);</pre>

DESCRIPTION Each of these functions computes an in-place multiplication of a vector by a scalar.

For real data, the following equation is used:

$$xz[i] = xz[i] * c[0]$$

where $i = 0, 1, \dots, (n - 1)$.

For complex data, the following equation is used:

$$\begin{aligned} tmp &= xz[2*i] \\ xz[2*i] &= tmp*c[0] - xz[2*i + 1]*c[1] \\ xz[2*i + 1] &= tmp*c[1] + xz[2*i + 1]*c[0] \end{aligned}$$

where $i = 0, 1, \dots, (n - 1)$.

PARAMETERS Each of the functions takes the following arguments:

- xz* Pointer to the first element of the source and destination vector.
- c* Pointer to the source scalar. When the function is used with complex data types, *c*[0] contains the real part of the scalar, and *c*[1] contains the imaginary part of the scalar.
- n* Number of elements in the vectors.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_VectorMulS_U8_U8_Mod(3MLIB)`, `attributes(5)`

mllib_VectorMulS_U8_U8_Mod(3MLIB)

NAME	mllib_VectorMulS_U8_U8_Mod, mllib_VectorMulS_U8_U8_Sat, mllib_VectorMulS_U8C_U8C_Mod, mllib_VectorMulS_U8C_U8C_Sat, mllib_VectorMulS_S8_S8_Mod, mllib_VectorMulS_S8_S8_Sat, mllib_VectorMulS_S8C_S8C_Mod, mllib_VectorMulS_S8C_S8C_Sat, mllib_VectorMulS_S16_U8_Mod, mllib_VectorMulS_S16_U8_Sat, mllib_VectorMulS_S16_S8_Mod, mllib_VectorMulS_S16_S8_Sat, mllib_VectorMulS_S16_S16_Mod, mllib_VectorMulS_S16_S16_Sat, mllib_VectorMulS_S16C_U8C_Mod, mllib_VectorMulS_S16C_U8C_Sat, mllib_VectorMulS_S16C_S8C_Mod, mllib_VectorMulS_S16C_S8C_Sat, mllib_VectorMulS_S16C_S16C_Mod, mllib_VectorMulS_S16C_S16C_Sat, mllib_VectorMulS_S32_S16_Mod, mllib_VectorMulS_S32_S16_Sat, mllib_VectorMulS_S32_S32_Mod, mllib_VectorMulS_S32_S32_Sat, mllib_VectorMulS_S32C_S16C_Mod, mllib_VectorMulS_S32C_S16C_Sat, mllib_VectorMulS_S32C_S32C_Mod, mllib_VectorMulS_S32C_S32C_Sat – vector multiplication by scalar
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VectorMulS_U8_U8_Mod(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *c, mllib_s32 n); mllib_status mllib_VectorMulS_U8_U8_Sat(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *c, mllib_s32 n); mllib_status mllib_VectorMulS_U8C_U8C_Mod(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *c, mllib_s32 n); mllib_status mllib_VectorMulS_U8C_U8C_Sat(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *c, mllib_s32 n); mllib_status mllib_VectorMulS_S8_S8_Mod(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *c, mllib_s32 n); mllib_status mllib_VectorMulS_S8_S8_Sat(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *c, mllib_s32 n); mllib_status mllib_VectorMulS_S8C_S8C_Mod(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *c, mllib_s32 n); mllib_status mllib_VectorMulS_S8C_S8C_Sat(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *c, mllib_s32 n); mllib_status mllib_VectorMulS_S16_U8_Mod(mllib_s16 *z, const mllib_u8 *x, const mllib_u8 *c, mllib_s32 n); mllib_status mllib_VectorMulS_S16_U8_Sat(mllib_s16 *z, const mllib_u8 *x, const mllib_u8 *c, mllib_s32 n); mllib_status mllib_VectorMulS_S16_S8_Mod(mllib_s16 *z, const mllib_s8 *x, const mllib_s8 *c, mllib_s32 n); mllib_status mllib_VectorMulS_S16_S8_Sat(mllib_s16 *z, const mllib_s8 *x, const mllib_s8 *c, mllib_s32 n);</pre>

mllib_VectorMulS_U8_U8_Mod(3MLIB)

```

mllib_status mllib_VectorMulS_S16_S16_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 n);
mllib_status mllib_VectorMulS_S16_S16_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 n);
mllib_status mllib_VectorMulS_S16C_U8C_Mod(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *c, mllib_s32 n);
mllib_status mllib_VectorMulS_S16C_U8C_Sat(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *c, mllib_s32 n);
mllib_status mllib_VectorMulS_S16C_S8C_Mod(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *c, mllib_s32 n);
mllib_status mllib_VectorMulS_S16C_S8C_Sat(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *c, mllib_s32 n);
mllib_status mllib_VectorMulS_S16C_S16C_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 n);
mllib_status mllib_VectorMulS_S16C_S16C_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 n);
mllib_status mllib_VectorMulS_S32_S16_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 n);
mllib_status mllib_VectorMulS_S32_S16_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 n);
mllib_status mllib_VectorMulS_S32_S32_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *c, mllib_s32 n);
mllib_status mllib_VectorMulS_S32_S32_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *c, mllib_s32 n);
mllib_status mllib_VectorMulS_S32C_S16C_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 n);
mllib_status mllib_VectorMulS_S32C_S16C_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 n);
mllib_status mllib_VectorMulS_S32C_S32C_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *c, mllib_s32 n);
mllib_status mllib_VectorMulS_S32C_S32C_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *c, mllib_s32 n);

```

DESCRIPTION

Each of these functions multiplies a vector by a scalar.

For real data, the following equation is used:

$$z[i] = x[i] * c[0]$$

where $i = 0, 1, \dots, (n - 1)$.

For complex data, the following equation is used:

mllib_VectorMulS_U8_U8_Mod(3MLIB)

$$\begin{aligned}z[2*i] &= x[2*i]*c[0] - x[2*i + 1]*c[1] \\z[2*i + 1] &= x[2*i]*c[1] + x[2*i + 1]*c[0]\end{aligned}$$

where $i = 0, 1, \dots, (n - 1)$.

PARAMETERS Each of the functions takes the following arguments:

z Pointer to the first element of the destination vector.
 x Pointer to the first element of the source vector.
 c Pointer to the source scalar. When the function is used with complex data types, $c[0]$ contains the real part of the scalar, and $c[1]$ contains the imaginary part of the scalar.
 n Number of elements in the vectors.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_VectorMulS_U8_U8_Mod(3MLIB)`, `attributes(5)`

NAME	mlib_VectorMul_U8_Mod, mlib_VectorMul_U8_Sat, mlib_VectorMul_U8C_Mod, mlib_VectorMul_U8C_Sat, mlib_VectorMul_S8_Mod, mlib_VectorMul_S8_Sat, mlib_VectorMul_S8C_Mod, mlib_VectorMul_S8C_Sat, mlib_VectorMul_S16_Mod, mlib_VectorMul_S16_Sat, mlib_VectorMul_S16C_Mod, mlib_VectorMul_S16C_Sat, mlib_VectorMul_S32_Mod, mlib_VectorMul_S32_Sat, mlib_VectorMul_S32C_Mod, mlib_VectorMul_S32C_Sat – vector multiplication, in place
SYNOPSIS	<pre>cc [flag...] file... -lmlib [library...] #include <mlib.h> mlib_status mlib_VectorMul_U8_Mod(mlib_u8 *xz, const mlib_u8 *y, mlib_s32 n); mlib_status mlib_VectorMul_U8_Sat(mlib_u8 *xz, const mlib_u8 *y, mlib_s32 n); mlib_status mlib_VectorMul_U8C_Mod(mlib_u8 *xz, const mlib_u8 *y, mlib_s32 n); mlib_status mlib_VectorMul_U8C_Sat(mlib_u8 *xz, const mlib_u8 *y, mlib_s32 n); mlib_status mlib_VectorMul_S8_Mod(mlib_s8 *xz, const mlib_s8 *y, mlib_s32 n); mlib_status mlib_VectorMul_S8_Sat(mlib_s8 *xz, const mlib_s8 *y, mlib_s32 n); mlib_status mlib_VectorMul_S8C_Mod(mlib_s8 *xz, const mlib_s8 *y, mlib_s32 n); mlib_status mlib_VectorMul_S8C_Sat(mlib_s8 *xz, const mlib_s8 *y, mlib_s32 n); mlib_status mlib_VectorMul_S16_Mod(mlib_s16 *xz, const mlib_s16 *y, mlib_s32 n); mlib_status mlib_VectorMul_S16_Sat(mlib_s16 *xz, const mlib_s16 *y, mlib_s32 n); mlib_status mlib_VectorMul_S16C_Mod(mlib_s16 *xz, const mlib_s16 *y, mlib_s32 n); mlib_status mlib_VectorMul_S16C_Sat(mlib_s16 *xz, const mlib_s16 *y, mlib_s32 n); mlib_status mlib_VectorMul_S32_Mod(mlib_s32 *xz, const mlib_s32 *y, mlib_s32 n); mlib_status mlib_VectorMul_S32_Sat(mlib_s32 *xz, const mlib_s32 *y, mlib_s32 n); mlib_status mlib_VectorMul_S32C_Mod(mlib_s32 *xz, const mlib_s32 *y, mlib_s32 n); mlib_status mlib_VectorMul_S32C_Sat(mlib_s32 *xz, const mlib_s32 *y, mlib_s32 n);</pre>

mllib_VectorMul_U8_Mod(3MLIB)

DESCRIPTION Each of these functions performs an in-place multiplication of one vector by another vector.

For real data, the following equation is used:

$$xz[i] = xz[i] * y[i]$$

where $i = 0, 1, \dots, (n - 1)$.

For complex data, the following equation is used:

$$\begin{aligned} \text{tmp} &= xz[2*i] \\ xz[2*i] &= \text{tmp}*y[2*i] - xz[2*i + 1]*y[2*i + 1] \\ xz[2*i + 1] &= \text{tmp}*y[2*i + 1] + xz[2*i + 1]*y[2*i] \end{aligned}$$

where $i = 0, 1, \dots, (n - 1)$.

PARAMETERS Each of the functions takes the following arguments:

xz Pointer to the first element of the first source and destination vector.

y Pointer to the first element of the second source vector.

n Number of elements in the vectors.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_VectorMul_U8_U8_Mod(3MLIB)`, `attributes(5)`

NAME	<p>mlib_VectorMul_U8_U8_Mod, mlib_VectorMul_U8_U8_Sat, mlib_VectorMul_U8C_U8C_Mod, mlib_VectorMul_U8C_U8C_Sat, mlib_VectorMul_S8_S8_Mod, mlib_VectorMul_S8_S8_Sat, mlib_VectorMul_S8C_S8C_Mod, mlib_VectorMul_S8C_S8C_Sat, mlib_VectorMul_S16_U8_Mod, mlib_VectorMul_S16_U8_Sat, mlib_VectorMul_S16_S8_Mod, mlib_VectorMul_S16_S8_Sat, mlib_VectorMul_S16_S16_Mod, mlib_VectorMul_S16_S16_Sat, mlib_VectorMul_S16C_U8C_Mod, mlib_VectorMul_S16C_U8C_Sat, mlib_VectorMul_S16C_S8C_Mod, mlib_VectorMul_S16C_S8C_Sat, mlib_VectorMul_S16C_S16C_Mod, mlib_VectorMul_S16C_S16C_Sat, mlib_VectorMul_S32_S16_Mod, mlib_VectorMul_S32_S16_Sat, mlib_VectorMul_S32_S32_Mod, mlib_VectorMul_S32_S32_Sat, mlib_VectorMul_S32C_S16C_Mod, mlib_VectorMul_S32C_S16C_Sat, mlib_VectorMul_S32C_S32C_Mod, mlib_VectorMul_S32C_S32C_Sat – vector multiplication</p>
SYNOPSIS	<pre>cc [flag...] file... -lmlib [library...] #include <mlib.h> mlib_status mlib_VectorMul_U8_U8_Mod(mlib_u8 *z, const mlib_u8 *x, const mlib_u8 *y, mlib_s32 n); mlib_status mlib_VectorMul_U8_U8_Sat(mlib_u8 *z, const mlib_u8 *x, const mlib_u8 *y, mlib_s32 n); mlib_status mlib_VectorMul_U8C_U8C_Mod(mlib_u8 *z, const mlib_u8 *x, const mlib_u8 *y, mlib_s32 n); mlib_status mlib_VectorMul_U8C_U8C_Sat(mlib_u8 *z, const mlib_u8 *x, const mlib_u8 *y, mlib_s32 n); mlib_status mlib_VectorMul_S8_S8_Mod(mlib_s8 *z, const mlib_s8 *x, const mlib_s8 *y, mlib_s32 n); mlib_status mlib_VectorMul_S8_S8_Sat(mlib_s8 *z, const mlib_s8 *x, const mlib_s8 *y, mlib_s32 n); mlib_status mlib_VectorMul_S8C_S8C_Mod(mlib_s8 *z, const mlib_s8 *x, const mlib_s8 *y, mlib_s32 n); mlib_status mlib_VectorMul_S8C_S8C_Sat(mlib_s8 *z, const mlib_s8 *x, const mlib_s8 *y, mlib_s32 n); mlib_status mlib_VectorMul_S16_U8_Mod(mlib_s16 *z, const mlib_u8 *x, const mlib_u8 *y, mlib_s32 n); mlib_status mlib_VectorMul_S16_U8_Sat(mlib_s16 *z, const mlib_u8 *x, const mlib_u8 *y, mlib_s32 n); mlib_status mlib_VectorMul_S16_S8_Mod(mlib_s16 *z, const mlib_s8 *x, const mlib_s8 *y, mlib_s32 n); mlib_status mlib_VectorMul_S16_S8_Sat(mlib_s16 *z, const mlib_s8 *x, const mlib_s8 *y, mlib_s32 n);</pre>

mllib_VectorMul_U8_U8_Mod(3MLIB)

```
mllib_status mllib_VectorMul_S16_S16_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 n);
mllib_status mllib_VectorMul_S16_S16_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 n);
mllib_status mllib_VectorMul_S16C_U8C_Mod(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *y, mllib_s32 n);
mllib_status mllib_VectorMul_S16C_U8C_Sat(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *y, mllib_s32 n);
mllib_status mllib_VectorMul_S16C_S8C_Mod(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *y, mllib_s32 n);
mllib_status mllib_VectorMul_S16C_S8C_Sat(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *y, mllib_s32 n);
mllib_status mllib_VectorMul_S16C_S16C_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 n);
mllib_status mllib_VectorMul_S16C_S16C_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 n);
mllib_status mllib_VectorMul_S32_S16_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 n);
mllib_status mllib_VectorMul_S32_S16_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 n);
mllib_status mllib_VectorMul_S32_S32_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 n);
mllib_status mllib_VectorMul_S32_S32_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 n);
mllib_status mllib_VectorMul_S32C_S16C_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 n);
mllib_status mllib_VectorMul_S32C_S16C_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 n);
mllib_status mllib_VectorMul_S32C_S32C_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 n);
mllib_status mllib_VectorMul_S32C_S32C_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 n);
```

DESCRIPTION

Each of these functions multiplies one vector by another vector.

For real data, the following equation is used:

$$z[i] = x[i] * y[i]$$

where $i = 0, 1, \dots, (n - 1)$.

For complex data, the following equation is used:

mllib_VectorMul_U8_U8_Mod(3MLIB)

$$\begin{aligned}z[2*i] &= x[2*i]*y[2*i] - x[2*i + 1]*y[2*i + 1] \\z[2*i + 1] &= x[2*i]*y[2*i + 1] + x[2*i + 1]*y[2*i]\end{aligned}$$

where $i = 0, 1, \dots, (n - 1)$.

PARAMETERS Each of the functions takes the following arguments:

- z* Pointer to the first element of the destination vector.
- x* Pointer to the first element of the first source vector.
- y* Pointer to the first element of the second source vector.
- n* Number of elements in the vectors.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_VectorMul_U8_Mod(3MLIB)`, `attributes(5)`

mllib_VectorNorm_U8_Sat(3MLIB)

NAME	mllib_VectorNorm_U8_Sat, mllib_VectorNorm_S8_Sat, mllib_VectorNorm_S16_Sat, mllib_VectorNorm_S32_Sat – vector norm						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VectorNorm_U8_Sat(mllib_d64 *z, const mllib_u8 *x, mllib_s32 n); mllib_status mllib_VectorNorm_S8_Sat(mllib_d64 *z, const mllib_s8 *x, mllib_s32 n); mllib_status mllib_VectorNorm_S16_Sat(mllib_d64 *z, const mllib_s16 *x, mllib_s32 n); mllib_status mllib_VectorNorm_S32_Sat(mllib_d64 *z, const mllib_s32 *x, mllib_s32 n);</pre>						
DESCRIPTION	<p>Each of these functions computes the vector normal.</p> <p>The following equation is used:</p> $z[0] = \left(\sum_{i=0}^{n-1} x[i]**2 \right)**0.5$						
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <p><i>z</i> Pointer to the norm of the vector.</p> <p><i>x</i> Pointer to the first element of the source vector.</p> <p><i>n</i> Number of elements in the vectors.</p>						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	attributes(5)						

mllib_VectorReverseByteOrder(3MLIB)

NAME mllib_VectorReverseByteOrder – reverse byte order of vector

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VectorReverseByteOrder(void *z, const void *x,
      mllib_s32 n, mllib_s32 s);
```

DESCRIPTION The `mllib_VectorReverseByteOrder()` function changes the encoding of each element from big endian to little endian, or from little endian to big endian.

It copies and reverses the byte order of each element of the input vector into the output vector.

PARAMETERS The function takes the following arguments:

- z* Pointer to the output vector.
- x* Pointer to the input vector.
- n* Number of elements in the vectors.
- s* Size of elements in bytes.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_VectorReverseByteOrder_Inp\(3MLIB\)](#),
[mllib_VectorReverseByteOrder_S16\(3MLIB\)](#),
[mllib_VectorReverseByteOrder_S16_S16\(3MLIB\)](#), `attributes(5)`

mllib_VectorReverseByteOrder_Inp(3MLIB)

NAME | mllib_VectorReverseByteOrder_Inp – reverse byte order of vector, in place

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VectorReverseByteOrder_Inp(void *xz, mllib_s32 n,
      mllib_s32 s);
```

DESCRIPTION | The mllib_VectorReverseByteOrder_Inp() function changes the encoding of each element from big endian to little endian, or from little endian to big endian.

It reverses the byte order of each element of the vector, in place.

PARAMETERS | The function takes the following arguments:

xz | Pointer to the input and output vector.

n | Number of elements in the vectors.

s | Size of elements in bytes.

RETURN VALUES | The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_VectorReverseByteOrder(3MLIB),
mllib_VectorReverseByteOrder_S16(3MLIB),
mllib_VectorReverseByteOrder_S16_S16(3MLIB), attributes(5)

mllib_VectorReverseByteOrder_S16(3MLIB)

NAME | mllib_VectorReverseByteOrder_S16, mllib_VectorReverseByteOrder_U16, mllib_VectorReverseByteOrder_S32, mllib_VectorReverseByteOrder_U32, mllib_VectorReverseByteOrder_S64, mllib_VectorReverseByteOrder_U64, mllib_VectorReverseByteOrder_F32, mllib_VectorReverseByteOrder_D64 – reverse byte order of vector, in place

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VectorReverseByteOrder_S16(mllib_s16 *xz,
        mllib_s32 n);

mllib_status mllib_VectorReverseByteOrder_U16(mllib_u16 *xz,
        mllib_s32 n);

mllib_status mllib_VectorReverseByteOrder_S32(mllib_s32 *xz,
        mllib_s32 n);

mllib_status mllib_VectorReverseByteOrder_U32(mllib_u32 *xz,
        mllib_s32 n);

mllib_status mllib_VectorReverseByteOrder_S64(mllib_s64 *xz,
        mllib_s32 n);

mllib_status mllib_VectorReverseByteOrder_U64(mllib_u64 *xz,
        mllib_s32 n);

mllib_status mllib_VectorReverseByteOrder_F32(mllib_f32 *xz,
        mllib_s32 n);

mllib_status mllib_VectorReverseByteOrder_D64(mllib_d64 *xz,
        mllib_s32 n);
```

DESCRIPTION | Each of these functions changes the encoding of each element from big endian to little endian, or from little endian to big endian.

It reverses the byte order of each element of the vector, in place.

PARAMETERS | Each of the functions takes the following arguments:

- xz* Pointer to input and output vector.
- n* Number of elements in the vectors.

RETURN VALUES | Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

mllib_VectorReverseByteOrder_S16(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	MT-Safe

SEE ALSO [mllib_VectorReverseByteOrder\(3MLIB\)](#),
[mllib_VectorReverseByteOrder_Inp\(3MLIB\)](#),
[mllib_VectorReverseByteOrder_S16_S16\(3MLIB\)](#), [attributes\(5\)](#)

mllib_VectorReverseByteOrder_S16_S16(3MLIB)

NAME	mllib_VectorReverseByteOrder_S16_S16, mllib_VectorReverseByteOrder_U16_U16, mllib_VectorReverseByteOrder_S32_S32, mllib_VectorReverseByteOrder_U32_U32, mllib_VectorReverseByteOrder_S64_S64, mllib_VectorReverseByteOrder_U64_U64, mllib_VectorReverseByteOrder_F32_F32, mllib_VectorReverseByteOrder_D64_D64 – reverse byte order of vector
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_VectorReverseByteOrder_S16_S16(mllib_s16 *z, const mllib_s16 *x, mllib_s32 n); mllib_status mllib_VectorReverseByteOrder_U16_U16(mllib_u16 *z, const mllib_u16 *x, mllib_s32 n); mllib_status mllib_VectorReverseByteOrder_S32_S32(mllib_s32 *z, const mllib_s32 *x, mllib_s32 n); mllib_status mllib_VectorReverseByteOrder_U32_U32(mllib_u32 *z, const mllib_u32 *x, mllib_s32 n); mllib_status mllib_VectorReverseByteOrder_S64_S64(mllib_s64 *z, const mllib_s64 *x, mllib_s32 n); mllib_status mllib_VectorReverseByteOrder_U64_U64(mllib_u64 *z, const mllib_u64 *x, mllib_s32 n); mllib_status mllib_VectorReverseByteOrder_F32_F32(mllib_f32 *z, const mllib_f32 *x, mllib_s32 n); mllib_status mllib_VectorReverseByteOrder_D64_D64(mllib_d64 *z, const mllib_d64 *x, mllib_s32 n);</pre>
DESCRIPTION	<p>Each of these functions changes the encoding of each element from big endian to little endian, or from little endian to big endian.</p> <p>It copies and reverses the byte order of each element of the input vector into the output vector.</p>
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <p><i>z</i> Pointer to the output vector.</p> <p><i>x</i> Pointer to input vector.</p> <p><i>n</i> Number of elements in the vectors.</p>
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:

`mlib_VectorReverseByteOrder_S16_S16(3MLIB)`

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

`mlib_VectorReverseByteOrder(3MLIB)`,
`mlib_VectorReverseByteOrder_Inp(3MLIB)`,
`mlib_VectorReverseByteOrder_S16(3MLIB)`, `attributes(5)`

NAME | mlib_VectorScale_U8_Mod, mlib_VectorScale_U8_Sat, mlib_VectorScale_U8C_Mod, mlib_VectorScale_U8C_Sat, mlib_VectorScale_S8_Mod, mlib_VectorScale_S8_Sat, mlib_VectorScale_S8C_Mod, mlib_VectorScale_S8C_Sat, mlib_VectorScale_S16_Mod, mlib_VectorScale_S16_Sat, mlib_VectorScale_S16C_Mod, mlib_VectorScale_S16C_Sat, mlib_VectorScale_S32_Mod, mlib_VectorScale_S32_Sat, mlib_VectorScale_S32C_Mod, mlib_VectorScale_S32C_Sat – vector linear scaling, in place

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_VectorScale_U8_Mod(mlib_u8 *xz, const mlib_u8
    *a, const mlib_u8 *b, mlib_s32 n);

mlib_status mlib_VectorScale_U8_Sat(mlib_u8 *xz, const mlib_u8
    *a, const mlib_u8 *b, mlib_s32 n);

mlib_status mlib_VectorScale_U8C_Mod(mlib_u8 *xz, const mlib_u8
    *a, const mlib_u8 *b, mlib_s32 n);

mlib_status mlib_VectorScale_U8C_Sat(mlib_u8 *xz, const mlib_u8
    *a, const mlib_u8 *b, mlib_s32 n);

mlib_status mlib_VectorScale_S8_Mod(mlib_s8 *xz, const mlib_s8
    *a, const mlib_s8 *b, mlib_s32 n);

mlib_status mlib_VectorScale_S8_Sat(mlib_s8 *xz, const mlib_s8
    *a, const mlib_s8 *b, mlib_s32 n);

mlib_status mlib_VectorScale_S8C_Mod(mlib_s8 *xz, const mlib_s8
    *a, const mlib_s8 *b, mlib_s32 n);

mlib_status mlib_VectorScale_S8C_Sat(mlib_s8 *xz, const mlib_s8
    *a, const mlib_s8 *b, mlib_s32 n);

mlib_status mlib_VectorScale_S16_Mod(mlib_s16 *xz, const mlib_s16
    *a, const mlib_s16 *b, mlib_s32 n);

mlib_status mlib_VectorScale_S16_Sat(mlib_s16 *xz, const mlib_s16
    *a, const mlib_s16 *b, mlib_s32 n);

mlib_status mlib_VectorScale_S16C_Mod(mlib_s16 *xz, const
    mlib_s16 *a, const mlib_s16 *b, mlib_s32 n);

mlib_status mlib_VectorScale_S16C_Sat(mlib_s16 *xz, const
    mlib_s16 *a, const mlib_s16 *b, mlib_s32 n);

mlib_status mlib_VectorScale_S32_Mod(mlib_s32 *xz, const mlib_s32
    *a, const mlib_s32 *b, mlib_s32 n);

mlib_status mlib_VectorScale_S32_Sat(mlib_s32 *xz, const mlib_s32
    *a, const mlib_s32 *b, mlib_s32 n);

mlib_status mlib_VectorScale_S32C_Mod(mlib_s32 *xz, const
    mlib_s32 *a, const mlib_s32 *b, mlib_s32 n);

mlib_status mlib_VectorScale_S32C_Sat(mlib_s32 *xz, const
    mlib_s32 *a, const mlib_s32 *b, mlib_s32 n);
```

mllib_VectorScale_U8_Mod(3MLIB)

DESCRIPTION Each of these functions performs an in-place scaling of a vector by multiplying by a scalar and adding an offset.

For real data, the following equation is used:

$$xz[i] = a[0]*xz[i] + b[0]$$

where $i = 0, 1, \dots, (n - 1)$.

For complex data, the following equation is used:

$$\begin{aligned} \text{tmp} &= xz[2*i] \\ xz[2*i] &= a[0]*\text{tmp} - a[1]*xz[2*i + 1] + b[0] \\ xz[2*i + 1] &= a[1]*\text{tmp} + a[0]*xz[2*i + 1] + b[1] \end{aligned}$$

where $i = 0, 1, \dots, (n - 1)$.

PARAMETERS Each of the functions takes the following arguments:

xz Pointer to the first element of the source and destination vector.

a Pointer to the source scaling factor. When the function is used with complex data types, *a*[0] contains the real part of the scaling factor, and *a*[1] contains the imaginary part of the scaling factor.

b Pointer to the source offset. When the function is used with complex data types, *b*[0] contains the real part of the offset, and *b*[1] contains the imaginary part of the offset.

n Number of elements in the vectors.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_VectorScale_U8_U8_Mod\(3MLIB\)](#), [attributes\(5\)](#)

mllib_VectorScale_U8_U8_Mod(3MLIB)

NAME | mllib_VectorScale_U8_U8_Mod, mllib_VectorScale_U8_U8_Sat,
 mllib_VectorScale_U8C_U8C_Mod, mllib_VectorScale_U8C_U8C_Sat,
 mllib_VectorScale_S8_S8_Mod, mllib_VectorScale_S8_S8_Sat,
 mllib_VectorScale_S8C_S8C_Mod, mllib_VectorScale_S8C_S8C_Sat,
 mllib_VectorScale_S16_U8_Mod, mllib_VectorScale_S16_U8_Sat,
 mllib_VectorScale_S16_S8_Mod, mllib_VectorScale_S16_S8_Sat,
 mllib_VectorScale_S16_S16_Mod, mllib_VectorScale_S16_S16_Sat,
 mllib_VectorScale_S16C_U8C_Mod, mllib_VectorScale_S16C_U8C_Sat,
 mllib_VectorScale_S16C_S8C_Mod, mllib_VectorScale_S16C_S8C_Sat,
 mllib_VectorScale_S16C_S16C_Mod, mllib_VectorScale_S16C_S16C_Sat,
 mllib_VectorScale_S32_S16_Mod, mllib_VectorScale_S32_S16_Sat,
 mllib_VectorScale_S32_S32_Mod, mllib_VectorScale_S32_S32_Sat,
 mllib_VectorScale_S32C_S16C_Mod, mllib_VectorScale_S32C_S16C_Sat,
 mllib_VectorScale_S32C_S32C_Mod, mllib_VectorScale_S32C_S32C_Sat – vector linear scaling

SYNOPSIS | `cc [flag...] file... -lmllib [library...]
 #include <mllib.h>`

```

mllib_status mllib_VectorScale_U8_U8_Mod(mllib_u8 *z, const mllib_u8
    *x, const mllib_u8 *a, const mllib_u8 *b, mllib_s32 n);

mllib_status mllib_VectorScale_U8_U8_Sat(mllib_u8 *z, const mllib_u8
    *x, const mllib_u8 *a, const mllib_u8 *b, mllib_s32 n);

mllib_status mllib_VectorScale_U8C_U8C_Mod(mllib_u8 *z, const
    mllib_u8 *x, const mllib_u8 *a, const mllib_u8 *b, mllib_s32 n);

mllib_status mllib_VectorScale_U8C_U8C_Sat(mllib_u8 *z, const
    mllib_u8 *x, const mllib_u8 *a, const mllib_u8 *b, mllib_s32 n);

mllib_status mllib_VectorScale_S8_S8_Mod(mllib_s8 *z, const mllib_s8
    *x, const mllib_s8 *a, const mllib_s8 *b, mllib_s32 n);

mllib_status mllib_VectorScale_S8_S8_Sat(mllib_s8 *z, const mllib_s8
    *x, const mllib_s8 *a, const mllib_s8 *b, mllib_s32 n);

mllib_status mllib_VectorScale_S8C_S8C_Mod(mllib_s8 *z, const
    mllib_s8 *x, const mllib_s8 *a, const mllib_s8 *b, mllib_s32 n);

mllib_status mllib_VectorScale_S8C_S8C_Sat(mllib_s8 *z, const
    mllib_s8 *x, const mllib_s8 *a, const mllib_s8 *b, mllib_s32 n);

mllib_status mllib_VectorScale_S16_U8_Mod(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *a, const mllib_u8 *b, mllib_s32 n);

mllib_status mllib_VectorScale_S16_U8_Sat(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *a, const mllib_u8 *b, mllib_s32 n);

mllib_status mllib_VectorScale_S16_S8_Mod(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *a, const mllib_s8 *b, mllib_s32 n);

mllib_status mllib_VectorScale_S16_S8_Sat(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *a, const mllib_s8 *b, mllib_s32 n);
  
```

mllib_VectorScale_U8_U8_Mod(3MLIB)

```
mllib_status mllib_VectorScale_S16_S16_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *a, const mllib_s16 *b, mllib_s32
    n);

mllib_status mllib_VectorScale_S16_S16_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *a, const mllib_s16 *b, mllib_s32
    n);

mllib_status mllib_VectorScale_S16C_U8C_Mod(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *a, const mllib_u8 *b, mllib_s32 n);

mllib_status mllib_VectorScale_S16C_U8C_Sat(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *a, const mllib_u8 *b, mllib_s32 n);

mllib_status mllib_VectorScale_S16C_S8C_Mod(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *a, const mllib_s8 *b, mllib_s32 n);

mllib_status mllib_VectorScale_S16C_S8C_Sat(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *a, const mllib_s8 *b, mllib_s32 n);

mllib_status mllib_VectorScale_S16C_S16C_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *a, const mllib_s16 *b, mllib_s32
    n);

mllib_status mllib_VectorScale_S16C_S16C_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *a, const mllib_s16 *b, mllib_s32
    n);

mllib_status mllib_VectorScale_S32_S16_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *a, const mllib_s16 *b, mllib_s32
    n);

mllib_status mllib_VectorScale_S32_S16_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *a, const mllib_s16 *b, mllib_s32
    n);

mllib_status mllib_VectorScale_S32_S32_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *a, const mllib_s32 *b, mllib_s32
    n);

mllib_status mllib_VectorScale_S32_S32_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *a, const mllib_s32 *b, mllib_s32
    n);

mllib_status mllib_VectorScale_S32C_S16C_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *a, const mllib_s16 *b, mllib_s32
    n);

mllib_status mllib_VectorScale_S32C_S16C_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *a, const mllib_s16 *b, mllib_s32
    n);

mllib_status mllib_VectorScale_S32C_S32C_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *a, const mllib_s32 *b, mllib_s32
    n);
```

mllib_VectorScale_U8_U8_Mod(3MLIB)

```
mllib_status mllib_VectorScale_S32C_S32C_Sat(mllib_s32 *z, const
mllib_s32 *x, const mllib_s32 *a, const mllib_s32 *b, mllib_s32
n);
```

DESCRIPTION Each of these functions scales a vector by multiplying by a scalar and adding an offset.

For real data, the following equation is used:

$$z[i] = a[0]*x[i] + b[0]$$

where $i = 0, 1, \dots, (n - 1)$.

For complex data, the following equation is used:

$$\begin{aligned} z[2*i] &= a[0]*x[2*i] - a[1]*x[2*i + 1] + b[0] \\ z[2*i + 1] &= a[1]*x[2*i] + a[0]*x[2*i + 1] + b[1] \end{aligned}$$

where $i = 0, 1, \dots, (n - 1)$.

PARAMETERS Each of the functions takes the following arguments:

- z* Pointer to the first element of the destination vector.
- x* Pointer to the first element of the source vector.
- a* Pointer to the source scaling factor. When the function is used with complex data types, *a*[0] contains the real part of the scaling factor, and *a*[1] contains the imaginary part of the scaling factor.
- b* Pointer to the source offset. When the function is used with complex data types, *b*[0] contains the real part of the offset, and *b*[1] contains the imaginary part of the offset.
- n* Number of elements in the vectors.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_VectorScale_U8_Mod(3MLIB)`, `attributes(5)`

mllib_VectorSet_U8(3MLIB)

NAME	mllib_VectorSet_U8, mllib_VectorSet_U8C, mllib_VectorSet_S8, mllib_VectorSet_S8C, mllib_VectorSet_S16, mllib_VectorSet_S16C, mllib_VectorSet_S32, mllib_VectorSet_S32C – set vector to specified value
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VectorSet_U8(mllib_u8 *z, const mllib_u8 *c, mllib_s32 n); mllib_status mllib_VectorSet_U8C(mllib_u8 *z, const mllib_u8 *c, mllib_s32 n); mllib_status mllib_VectorSet_S8(mllib_s8 *z, const mllib_s8 *c, mllib_s32 n); mllib_status mllib_VectorSet_S8C(mllib_s8 *z, const mllib_s8 *c, mllib_s32 n); mllib_status mllib_VectorSet_S16(mllib_s16 *z, const mllib_s16 *c, mllib_s32 n); mllib_status mllib_VectorSet_S16C(mllib_s16 *z, const mllib_s16 *c, mllib_s32 n); mllib_status mllib_VectorSet_S32(mllib_s32 *z, const mllib_s32 *c, mllib_s32 n); mllib_status mllib_VectorSet_S32C(mllib_s32 *z, const mllib_s32 *c, mllib_s32 n);</pre>
DESCRIPTION	<p>Each of these functions sets a vector to a specified value.</p> <p>For real data, the following equation is used:</p> $z[i] = c[0]$ <p>where $i = 0, 1, \dots, (n - 1)$.</p> <p>For complex data, the following equation is used:</p> $\begin{aligned} z[2*i] &= c[0] \\ z[2*i + 1] &= c[1] \end{aligned}$ <p>where $i = 0, 1, \dots, (n - 1)$.</p>
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <p><i>z</i> Pointer to the first element of the destination vector.</p> <p><i>c</i> Pointer to the source scalar. When the function is used with complex data types, <i>c</i> [0] contains the scaling factor for the real part, and <i>c</i> [1] contains the scaling factor for the imaginary part.</p> <p><i>n</i> Number of elements in the vector.</p>

mllib_VectorSet_U8(3MLIB)

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `attributes(5)`

mllib_VectorSplit_U8_U8C(3MLIB)

NAME	mllib_VectorSplit_U8_U8C, mllib_VectorSplit_S8_S8C, mllib_VectorSplit_S16_S16C, mllib_VectorSplit_S32_S32C – vector split						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VectorSplit_U8_U8C(mllib_u8 *r, mllib_u8 *i, const mllib_u8 *x, mllib_s32 n); mllib_status mllib_VectorSplit_S8_S8C(mllib_s8 *r, mllib_s8 *i, const mllib_s8 *x, mllib_s32 n); mllib_status mllib_VectorSplit_S16_S16C(mllib_s16 *r, mllib_s16 *i, const mllib_s16 *x, mllib_s32 n); mllib_status mllib_VectorSplit_S32_S32C(mllib_s32 *r, mllib_s32 *i, const mllib_s32 *x, mllib_s32 n);</pre>						
DESCRIPTION	<p>Each of these functions splits a complex vector into separate vectors containing the real and imaginary parts.</p> <p>The following equation is used:</p> $r[k] = z[2*k]$ $i[k] = z[2*k + 1]$ <p>where $k = 0, 1, \dots, (n - 1)$.</p>						
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <p><i>r</i> Pointer to the first element of the real part.</p> <p><i>i</i> Pointer to the first element of the imaginary part.</p> <p><i>x</i> Pointer to the first complex element of the source vector. $x[2*k]$ contains the real part, and $x[2*k + 1]$ contains the imaginary part.</p> <p><i>n</i> Number of elements in the vectors.</p>						
RETURN VALUES	Each of the functions returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_VectorMerge_U8C_U8(3MLIB) , attributes(5)						

NAME | mllib_VectorSubS_U8_Mod, mllib_VectorSubS_U8_Sat, mllib_VectorSubS_U8C_Mod, mllib_VectorSubS_U8C_Sat, mllib_VectorSubS_S8_Mod, mllib_VectorSubS_S8_Sat, mllib_VectorSubS_S8C_Mod, mllib_VectorSubS_S8C_Sat, mllib_VectorSubS_S16_Mod, mllib_VectorSubS_S16_Sat, mllib_VectorSubS_S16C_Mod, mllib_VectorSubS_S16C_Sat, mllib_VectorSubS_S32_Mod, mllib_VectorSubS_S32_Sat, mllib_VectorSubS_S32C_Mod, mllib_VectorSubS_S32C_Sat – vector subtraction from scalar, in place

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VectorSubS_U8_Mod(mllib_u8 *xz, const mllib_u8 *c,
mllib_s32 n);

mllib_status mllib_VectorSubS_U8_Sat(mllib_u8 *xz, const mllib_u8 *c,
mllib_s32 n);

mllib_status mllib_VectorSubS_U8C_Mod(mllib_u8 *xz, const mllib_u8
*c, mllib_s32 n);

mllib_status mllib_VectorSubS_U8C_Sat(mllib_u8 *xz, const mllib_u8
*c, mllib_s32 n);

mllib_status mllib_VectorSubS_S8_Mod(mllib_s8 *xz, const mllib_s8 *c,
mllib_s32 n);

mllib_status mllib_VectorSubS_S8_Sat(mllib_s8 *xz, const mllib_s8 *c,
mllib_s32 n);

mllib_status mllib_VectorSubS_S8C_Mod(mllib_s8 *xz, const mllib_s8
*c, mllib_s32 n);

mllib_status mllib_VectorSubS_S8C_Sat(mllib_s8 *xz, const mllib_s8
*c, mllib_s32 n);

mllib_status mllib_VectorSubS_S16_Mod(mllib_s16 *xz, const mllib_s16
*c, mllib_s32 n);

mllib_status mllib_VectorSubS_S16_Sat(mllib_s16 *xz, const mllib_s16
*c, mllib_s32 n);

mllib_status mllib_VectorSubS_S16C_Mod(mllib_s16 *xz, const mllib_s16
*c, mllib_s32 n);

mllib_status mllib_VectorSubS_S16C_Sat(mllib_s16 *xz, const mllib_s16
*c, mllib_s32 n);

mllib_status mllib_VectorSubS_S32_Mod(mllib_s32 *xz, const mllib_s32
*c, mllib_s32 n);

mllib_status mllib_VectorSubS_S32_Sat(mllib_s32 *xz, const mllib_s32
*c, mllib_s32 n);

mllib_status mllib_VectorSubS_S32C_Mod(mllib_s32 *xz, const mllib_s32
*c, mllib_s32 n);

mllib_status mllib_VectorSubS_S32C_Sat(mllib_s32 *xz, const mllib_s32
*c, mllib_s32 n);
```

mllib_VectorSubS_U8_Mod(3MLIB)

DESCRIPTION Each of these functions performs an in-place subtraction of a vector from a scalar.

For real data, the following equation is used:

$$xz[i] = c[0] - xz[i]$$

where $i = 0, 1, \dots, (n - 1)$.

For complex data, the following equation is used:

$$\begin{aligned}xz[2*i] &= c[0] - xz[2*i] \\xz[2*i + 1] &= c[1] - xz[2*i + 1]\end{aligned}$$

where $i = 0, 1, \dots, (n - 1)$.

PARAMETERS Each of the functions takes the following arguments:

xz Pointer to the first element of the source and destination vector.
c Pointer to the source scalar. When the function is used with complex data types, *c*[0] contains the scalar for the real part, and *c*[1] contains the scalar for the imaginary part.
n Number of elements in the vectors.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_VectorSubS_U8_U8_Mod\(3MLIB\)](#), [attributes\(5\)](#)

mllib_VectorSubS_U8_U8_Mod(3MLIB)

NAME | mllib_VectorSubS_U8_U8_Mod, mllib_VectorSubS_U8_U8_Sat,
 mllib_VectorSubS_U8C_U8C_Mod, mllib_VectorSubS_U8C_U8C_Sat,
 mllib_VectorSubS_S8_S8_Mod, mllib_VectorSubS_S8_S8_Sat,
 mllib_VectorSubS_S8C_S8C_Mod, mllib_VectorSubS_S8C_S8C_Sat,
 mllib_VectorSubS_S16_U8_Mod, mllib_VectorSubS_S16_U8_Sat,
 mllib_VectorSubS_S16_S8_Mod, mllib_VectorSubS_S16_S8_Sat,
 mllib_VectorSubS_S16_S16_Mod, mllib_VectorSubS_S16_S16_Sat,
 mllib_VectorSubS_S16C_U8C_Mod, mllib_VectorSubS_S16C_U8C_Sat,
 mllib_VectorSubS_S16C_S8C_Mod, mllib_VectorSubS_S16C_S8C_Sat,
 mllib_VectorSubS_S16C_S16C_Mod, mllib_VectorSubS_S16C_S16C_Sat,
 mllib_VectorSubS_S32_S16_Mod, mllib_VectorSubS_S32_S16_Sat,
 mllib_VectorSubS_S32_S32_Mod, mllib_VectorSubS_S32_S32_Sat,
 mllib_VectorSubS_S32C_S16C_Mod, mllib_VectorSubS_S32C_S16C_Sat,
 mllib_VectorSubS_S32C_S32C_Mod, mllib_VectorSubS_S32C_S32C_Sat – vector
 subtraction from scalar

SYNOPSIS | `cc [flag...] file... -lmllib [library...]`
`#include <mllib.h>`

`mllib_status mllib_VectorSubS_U8_U8_Mod(mllib_u8 *z, const mllib_u8
 *x, const mllib_u8 *c, mllib_s32 n);`

`mllib_status mllib_VectorSubS_U8_U8_Sat(mllib_u8 *z, const mllib_u8
 *x, const mllib_u8 *c, mllib_s32 n);`

`mllib_status mllib_VectorSubS_U8C_U8C_Mod(mllib_u8 *z, const mllib_u8
 *x, const mllib_u8 *c, mllib_s32 n);`

`mllib_status mllib_VectorSubS_U8C_U8C_Sat(mllib_u8 *z, const mllib_u8
 *x, const mllib_u8 *c, mllib_s32 n);`

`mllib_status mllib_VectorSubS_S8_S8_Mod(mllib_s8 *z, const mllib_s8
 *x, const mllib_s8 *c, mllib_s32 n);`

`mllib_status mllib_VectorSubS_S8_S8_Sat(mllib_s8 *z, const mllib_s8
 *x, const mllib_s8 *c, mllib_s32 n);`

`mllib_status mllib_VectorSubS_S8C_S8C_Mod(mllib_s8 *z, const mllib_s8
 *x, const mllib_s8 *c, mllib_s32 n);`

`mllib_status mllib_VectorSubS_S8C_S8C_Sat(mllib_s8 *z, const mllib_s8
 *x, const mllib_s8 *c, mllib_s32 n);`

`mllib_status mllib_VectorSubS_S16_U8_Mod(mllib_s16 *z, const mllib_u8
 *x, const mllib_u8 *c, mllib_s32 n);`

`mllib_status mllib_VectorSubS_S16_U8_Sat(mllib_s16 *z, const mllib_u8
 *x, const mllib_u8 *c, mllib_s32 n);`

`mllib_status mllib_VectorSubS_S16_S8_Mod(mllib_s16 *z, const mllib_s8
 *x, const mllib_s8 *c, mllib_s32 n);`

`mllib_status mllib_VectorSubS_S16_S8_Sat(mllib_s16 *z, const mllib_s8
 *x, const mllib_s8 *c, mllib_s32 n);`

mllib_VectorSubS_U8_U8_Mod(3MLIB)

```
mllib_status mllib_VectorSubS_S16_S16_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 n);
mllib_status mllib_VectorSubS_S16_S16_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 n);
mllib_status mllib_VectorSubS_S16C_U8C_Mod(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *c, mllib_s32 n);
mllib_status mllib_VectorSubS_S16C_U8C_Sat(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *c, mllib_s32 n);
mllib_status mllib_VectorSubS_S16C_S8C_Mod(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *c, mllib_s32 n);
mllib_status mllib_VectorSubS_S16C_S8C_Sat(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *c, mllib_s32 n);
mllib_status mllib_VectorSubS_S16C_S16C_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 n);
mllib_status mllib_VectorSubS_S16C_S16C_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 n);
mllib_status mllib_VectorSubS_S32_S16_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 n);
mllib_status mllib_VectorSubS_S32_S16_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 n);
mllib_status mllib_VectorSubS_S32_S32_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *c, mllib_s32 n);
mllib_status mllib_VectorSubS_S32_S32_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *c, mllib_s32 n);
mllib_status mllib_VectorSubS_S32C_S16C_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 n);
mllib_status mllib_VectorSubS_S32C_S16C_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *c, mllib_s32 n);
mllib_status mllib_VectorSubS_S32C_S32C_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *c, mllib_s32 n);
mllib_status mllib_VectorSubS_S32C_S32C_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *c, mllib_s32 n);
```

DESCRIPTION

Each of these functions subtracts a vector from a scalar.

For real data, the following equation is used:

$$z[i] = c[0] - x[i]$$

where $i = 0, 1, \dots, (n - 1)$.

For complex data, the following equation is used:

```
z[2*i]      = c[0] - x[2*i]
z[2*i + 1] = c[1] - x[2*i + 1]
```

where $i = 0, 1, \dots, (n - 1)$.

PARAMETERS Each of the functions takes the following arguments:

- z* Pointer to the first element of the destination vector.
- x* Pointer to the first element of the source vector.
- c* Pointer to the source scalar. When the function is used with complex data types, *c*[0] contains the scalar for the real part, and *c*[1] contains the scalar for the imaginary part.
- n* Number of elements in the vectors.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_VectorSubS_U8_U8_Mod(3MLIB)`, `attributes(5)`

mllib_VectorSub_U8_Mod(3MLIB)

NAME	mllib_VectorSub_U8_Mod, mllib_VectorSub_U8_Sat, mllib_VectorSub_U8C_Mod, mllib_VectorSub_U8C_Sat, mllib_VectorSub_S8_Mod, mllib_VectorSub_S8_Sat, mllib_VectorSub_S8C_Mod, mllib_VectorSub_S8C_Sat, mllib_VectorSub_S16_Mod, mllib_VectorSub_S16_Sat, mllib_VectorSub_S16C_Mod, mllib_VectorSub_S16C_Sat, mllib_VectorSub_S32_Mod, mllib_VectorSub_S32_Sat, mllib_VectorSub_S32C_Mod, mllib_VectorSub_S32C_Sat – vector subtraction, in place
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VectorSub_U8_Mod(mllib_u8 *xz, const mllib_u8 *y, mllib_s32 n); mllib_status mllib_VectorSub_U8_Sat(mllib_u8 *xz, const mllib_u8 *y, mllib_s32 n); mllib_status mllib_VectorSub_U8C_Mod(mllib_u8 *xz, const mllib_u8 *y, mllib_s32 n); mllib_status mllib_VectorSub_U8C_Sat(mllib_u8 *xz, const mllib_u8 *y, mllib_s32 n); mllib_status mllib_VectorSub_S8_Mod(mllib_s8 *xz, const mllib_s8 *y, mllib_s32 n); mllib_status mllib_VectorSub_S8_Sat(mllib_s8 *xz, const mllib_s8 *y, mllib_s32 n); mllib_status mllib_VectorSub_S8C_Mod(mllib_s8 *xz, const mllib_s8 *y, mllib_s32 n); mllib_status mllib_VectorSub_S8C_Sat(mllib_s8 *xz, const mllib_s8 *y, mllib_s32 n); mllib_status mllib_VectorSub_S16_Mod(mllib_s16 *xz, const mllib_s16 *y, mllib_s32 n); mllib_status mllib_VectorSub_S16_Sat(mllib_s16 *xz, const mllib_s16 *y, mllib_s32 n); mllib_status mllib_VectorSub_S16C_Mod(mllib_s16 *xz, const mllib_s16 *y, mllib_s32 n); mllib_status mllib_VectorSub_S16C_Sat(mllib_s16 *xz, const mllib_s16 *y, mllib_s32 n); mllib_status mllib_VectorSub_S32_Mod(mllib_s32 *xz, const mllib_s32 *y, mllib_s32 n); mllib_status mllib_VectorSub_S32_Sat(mllib_s32 *xz, const mllib_s32 *y, mllib_s32 n); mllib_status mllib_VectorSub_S32C_Mod(mllib_s32 *xz, const mllib_s32 *y, mllib_s32 n); mllib_status mllib_VectorSub_S32C_Sat(mllib_s32 *xz, const mllib_s32 *y, mllib_s32 n);</pre>

DESCRIPTION Each of these functions performs an in-place subtraction of a vector from another vector.

It uses the following equation:

$$xz[i] = xz[i] - y[i]$$

where $i = 0, 1, \dots, (n - 1)$ for real data; $i = 0, 1, \dots, (2*n - 1)$ for complex data.

PARAMETERS Each of the functions takes the following arguments:

xz Pointer to the first element of the first source and destination vector.

y Pointer to the first element of the second source vector.

n Number of elements in the vectors.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_VectorSub_U8_U8_Mod(3MLIB)`, `attributes(5)`

mllib_VectorSub_U8_U8_Mod(3MLIB)

NAME	mllib_VectorSub_U8_U8_Mod, mllib_VectorSub_U8_U8_Sat, mllib_VectorSub_U8C_U8C_Mod, mllib_VectorSub_U8C_U8C_Sat, mllib_VectorSub_S8_S8_Mod, mllib_VectorSub_S8_S8_Sat, mllib_VectorSub_S8C_S8C_Mod, mllib_VectorSub_S8C_S8C_Sat, mllib_VectorSub_S16_U8_Mod, mllib_VectorSub_S16_U8_Sat, mllib_VectorSub_S16_S8_Mod, mllib_VectorSub_S16_S8_Sat, mllib_VectorSub_S16_S16_Mod, mllib_VectorSub_S16_S16_Sat, mllib_VectorSub_S16C_U8C_Mod, mllib_VectorSub_S16C_U8C_Sat, mllib_VectorSub_S16C_S8C_Mod, mllib_VectorSub_S16C_S8C_Sat, mllib_VectorSub_S16C_S16C_Mod, mllib_VectorSub_S16C_S16C_Sat, mllib_VectorSub_S32_S16_Mod, mllib_VectorSub_S32_S16_Sat, mllib_VectorSub_S32_S32_Mod, mllib_VectorSub_S32_S32_Sat, mllib_VectorSub_S32C_S16C_Mod, mllib_VectorSub_S32C_S16C_Sat, mllib_VectorSub_S32C_S32C_Mod, mllib_VectorSub_S32C_S32C_Sat – vector subtraction
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VectorSub_U8_U8_Mod(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 n); mllib_status mllib_VectorSub_U8_U8_Sat(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 n); mllib_status mllib_VectorSub_U8C_U8C_Mod(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 n); mllib_status mllib_VectorSub_U8C_U8C_Sat(mllib_u8 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 n); mllib_status mllib_VectorSub_S8_S8_Mod(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 n); mllib_status mllib_VectorSub_S8_S8_Sat(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 n); mllib_status mllib_VectorSub_S8C_S8C_Mod(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 n); mllib_status mllib_VectorSub_S8C_S8C_Sat(mllib_s8 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 n); mllib_status mllib_VectorSub_S16_U8_Mod(mllib_s16 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 n); mllib_status mllib_VectorSub_S16_U8_Sat(mllib_s16 *z, const mllib_u8 *x, const mllib_u8 *y, mllib_s32 n); mllib_status mllib_VectorSub_S16_S8_Mod(mllib_s16 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 n); mllib_status mllib_VectorSub_S16_S8_Sat(mllib_s16 *z, const mllib_s8 *x, const mllib_s8 *y, mllib_s32 n);</pre>

mllib_VectorSub_U8_U8_Mod(3MLIB)

```

mllib_status mllib_VectorSub_S16_S16_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 n);
mllib_status mllib_VectorSub_S16_S16_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 n);
mllib_status mllib_VectorSub_S16C_U8C_Mod(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *y, mllib_s32 n);
mllib_status mllib_VectorSub_S16C_U8C_Sat(mllib_s16 *z, const
    mllib_u8 *x, const mllib_u8 *y, mllib_s32 n);
mllib_status mllib_VectorSub_S16C_S8C_Mod(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *y, mllib_s32 n);
mllib_status mllib_VectorSub_S16C_S8C_Sat(mllib_s16 *z, const
    mllib_s8 *x, const mllib_s8 *y, mllib_s32 n);
mllib_status mllib_VectorSub_S16C_S16C_Mod(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 n);
mllib_status mllib_VectorSub_S16C_S16C_Sat(mllib_s16 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 n);
mllib_status mllib_VectorSub_S32_S16_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 n);
mllib_status mllib_VectorSub_S32_S16_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 n);
mllib_status mllib_VectorSub_S32_S32_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 n);
mllib_status mllib_VectorSub_S32_S32_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 n);
mllib_status mllib_VectorSub_S32C_S16C_Mod(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 n);
mllib_status mllib_VectorSub_S32C_S16C_Sat(mllib_s32 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 n);
mllib_status mllib_VectorSub_S32C_S32C_Mod(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 n);
mllib_status mllib_VectorSub_S32C_S32C_Sat(mllib_s32 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 n);

```

DESCRIPTION

Each of these functions subtracts one vector from another vector.

It uses the following equation:

$$z[i] = x[i] - y[i]$$

where $i = 0, 1, \dots, (n - 1)$ for real data; $i = 0, 1, \dots, (2*n - 1)$ for complex data.

mllib_VectorSub_U8_U8_Mod(3MLIB)

PARAMETERS Each of the functions takes the following arguments:

z Pointer to the first element of the destination vector.
x Pointer to the first element of the first source vector.
y Pointer to the first element of the second source vector.
n Number of elements in the vectors.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_VectorSub_U8_U8_Mod\(3MLIB\)](#), [attributes\(5\)](#)

NAME mllib_VectorSumAbsDiff_U8_Sat, mllib_VectorSumAbsDiff_S8_Sat, mllib_VectorSumAbsDiff_S16_Sat, mllib_VectorSumAbsDiff_S32_Sat – sum of the absolute values of the differences of two vectors

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VectorSumAbsDiff_U8_Sat(mllib_d64 *z, const
    mllib_u8 *x, const mllib_u8 *y, mllib_s32 n);

mllib_status mllib_VectorSumAbsDiff_S8_Sat(mllib_d64 *z, const
    mllib_s8 *x, const mllib_s8 *y, mllib_s32 n);

mllib_status mllib_VectorSumAbsDiff_S16_Sat(mllib_d64 *z, const
    mllib_s16 *x, const mllib_s16 *y, mllib_s32 n);

mllib_status mllib_VectorSumAbsDiff_S32_Sat(mllib_d64 *z, const
    mllib_s32 *x, const mllib_s32 *y, mllib_s32 n);
```

DESCRIPTION Each of these functions computes the sum of the absolute values of the differences of two vectors.

The following equation is used:

$$z[0] = \sum_{i=0}^{n-1} |x[i] - y[i]|$$

PARAMETERS Each of the functions takes the following arguments:

z Pointer to the sum of the absolute differences between two vectors.

x Pointer to the first element of the first source vector.

y Pointer to the first element of the second source vector.

n Number of elements in the vectors.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO attributes(5)

mllib_VectorSumAbs_U8_Sat(3MLIB)

NAME	mllib_VectorSumAbs_U8_Sat, mllib_VectorSumAbs_S8_Sat, mllib_VectorSumAbs_S16_Sat, mllib_VectorSumAbs_S32_Sat – sum of the absolute values of a vector						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VectorSumAbs_U8_Sat(mllib_d64 *z, const mllib_u8 *x, mllib_s32 n); mllib_status mllib_VectorSumAbs_S8_Sat(mllib_d64 *z, const mllib_s8 *x, mllib_s32 n); mllib_status mllib_VectorSumAbs_S16_Sat(mllib_d64 *z, const mllib_s16 *x, mllib_s32 n); mllib_status mllib_VectorSumAbs_S32_Sat(mllib_d64 *z, const mllib_s32 *x, mllib_s32 n);</pre>						
DESCRIPTION	<p>Each of these functions computes the sum of the absolute values of a vector.</p> <p>The following equation is used:</p> $z[0] = \sum_{i=0}^{n-1} x[i] $						
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <p><i>z</i> Pointer to the sum of the absolute values of the vector.</p> <p><i>x</i> Pointer to the first element of the first source vector.</p> <p><i>n</i> Number of elements in the vectors.</p>						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	attributes(5)						

NAME mllib_VectorZero_U8, mllib_VectorZero_U8C, mllib_VectorZero_S8, mllib_VectorZero_S8C, mllib_VectorZero_S16, mllib_VectorZero_S16C, mllib_VectorZero_S32, mllib_VectorZero_S32C – initialize vector to zero

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VectorZero_U8(mllib_u8 *z, mllib_s32 n);
mllib_status mllib_VectorZero_U8C(mllib_u8 *z, mllib_s32 n);
mllib_status mllib_VectorZero_S8(mllib_s8 *z, mllib_s32 n);
mllib_status mllib_VectorZero_S8C(mllib_s8 *z, mllib_s32 n);
mllib_status mllib_VectorZero_S16(mllib_s16 *z, mllib_s32 n);
mllib_status mllib_VectorZero_S16C(mllib_s16 *z, mllib_s32 n);
mllib_status mllib_VectorZero_S32(mllib_s32 *z, mllib_s32 n);
mllib_status mllib_VectorZero_S32C(mllib_s32 *z, mllib_s32 n);
```

DESCRIPTION Each of these functions initializes a vector to zero.

The following equation is used:

$$z[i] = 0$$

where $i = 0, 1, \dots, (n - 1)$ for real data; $i = 0, 1, \dots, (2*n - 1)$ for complex data.

PARAMETERS Each of the functions takes the following arguments:

- z* Pointer to the first element of the destination vector.
- n* Number of elements in the vector.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO attributes(5)

mllib_version(3MLIB)

NAME mllib_version – return a version string

SYNOPSIS cc [*flag...*] *file...* -lmllib [*library...*]
#include <mllib.h>

```
char *mllib_version(void);
```

DESCRIPTION The mllib_version() function returns a string about the version of the library being used.

This function returns a string in the following format:

```
lib_name:version:build_date:target_isa
```

The *lib_name* is mediaLib. The *version* consists of four digits. The first two digits of the version are the major version. The third digit is the minor version, and the fourth digit is the micro version. The *build_date* is in the *yyyymmdd* format. The *target_isa* is the value used for the *-xarch=a* flag of the compiler when the library was built. For example, the following version string corresponds to a library in mediaLib version 2.1.0, which was built on 11/01/2001 and for the *sparcv8plus+vis* architecture.

```
mediaLib:0210:20011101:v8plusa
```

PARAMETERS The function takes no argument.

RETURN VALUES The function returns a pointer to a string of characters.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO attributes(5)

NAME | mllib_VideoAddBlock_U8_S16 – adds motion-compensated 8x8 block to the current block

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoAddBlock_U8_S16(mllib_u8 *curr_block, const
    mllib_s16 *mc_block, mllib_s32 stride);
```

DESCRIPTION | The mllib_VideoAddBlock_U8_S16() function performs additions of prediction and coefficient data. In other words, the function adds a motion-compensated 8x8 block to the current block. The stride applies to the current block.

PARAMETERS | The function takes the following arguments:

curr_block | Pointer to the current block. curr_block must be 8-byte aligned.

mc_block | Pointer to an 8x8 motion-compensated block (prediction data). mc_block must be 8-byte aligned.

stride | Stride, in bytes, between adjacent rows in the current block. stride must be a multiple of eight.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_VideoCopyRef_S16_U8(3MLIB),
mllib_VideoCopyRef_S16_U8_16x16(3MLIB),
mllib_VideoCopyRef_U8_U8_16x16(3MLIB),
mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB),
mllib_VideoH263OverlappedMC_S16_U8(3MLIB),
mllib_VideoH263OverlappedMC_U8_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveXY_U8_U8(3MLIB),
mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveY_U8_U8(3MLIB),
mllib_VideoInterpAveY_U8_U8_16x16(3MLIB),
mllib_VideoInterpX_S16_U8(3MLIB),
mllib_VideoInterpX_S16_U8_16x16(3MLIB),
mllib_VideoInterpX_U8_U8(3MLIB), mllib_VideoInterpXY_S16_U8(3MLIB),
mllib_VideoInterpXY_S16_U8_16x16(3MLIB),
mllib_VideoInterpXY_U8_U8(3MLIB),
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),

mllib_VideoAddBlock_U8_S16(3MLIB)

```
mllib_VideoInterpY_S16_U8(3MLIB),  
mllib_VideoInterpY_S16_U8_16x16(3MLIB),  
mllib_VideoInterpY_U8_U8(3MLIB),  
mllib_VideoInterpY_U8_U8_16x16(3MLIB),  
mllib_VideoP64Decimate_U8_U8(3MLIB),  
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),  
attributes(5)
```

mllib_VideoColorABGR2JFIFYCC420(3MLIB)

NAME | mllib_VideoColorABGR2JFIFYCC420 – ABGR to JFIF YCbCr color conversion

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorABGR2JFIFYCC420(mllib_u8 *y0, mllib_u8
    *y1, mllib_u8 *cb, mllib_u8 *cr, const mllib_u8 *abgr0, const
    mllib_u8 *abgr1, mllib_s32 n);
```

DESCRIPTION | The mllib_VideoColorABGR2JFIFYCC420() function performs color space conversion from ABGR to YCbCr to together with sampling rate conversion when used in the JPEG File Interchange Format (JFIF).

PARAMETERS | The function takes the following arguments:

<i>y0</i>	Pointer to upper destination Y component row. <i>y0</i> must be 8-byte aligned.
<i>y1</i>	Pointer to lower destination Y component row. <i>y1</i> must be 8-byte aligned.
<i>cb</i>	Pointer to destination Cb component row. <i>cb</i> must be 8-byte aligned.
<i>cr</i>	Pointer to destination Cr component row. <i>cr</i> must be 8-byte aligned.
<i>abgr0</i>	Pointer to upper source ABGR multi-component row. <i>abgr0</i> must be 8-byte aligned.
<i>abgr1</i>	Pointer to lower source ABGR multi-component row. <i>abgr1</i> must be 8-byte aligned.
<i>n</i>	Length of Y component row. <i>n</i> must be even. The length of Cb and Cr component rows must be <i>n</i> /2. The length of the RGB multi-component row must be 3* <i>n</i> .

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_VideoColorABGR2JFIFYCC422\(3MLIB\)](#),
[mllib_VideoColorARGB2JFIFYCC420\(3MLIB\)](#),
[mllib_VideoColorARGB2JFIFYCC422\(3MLIB\)](#),
[mllib_VideoColorRGB2JFIFYCC420\(3MLIB\)](#),
[mllib_VideoColorRGB2JFIFYCC422\(3MLIB\)](#), [attributes\(5\)](#)

mllib_VideoColorABGR2JFIFYCC422(3MLIB)

NAME	mllib_VideoColorABGR2JFIFYCC422 – ABGR to JFIF YCbCr color conversion										
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoColorABGR2JFIFYCC422 (mllib_u8 *y, mllib_u8 *cb, mllib_u8 *cr, const mllib_u8 *abgr, mllib_s32 n);</pre>										
DESCRIPTION	The <code>mllib_VideoColorABGR2JFIFYCC422()</code> function performs color space conversion from ABGR to YCbCr to together with sampling rate conversion when used in the JPEG File Interchange Format (JFIF).										
PARAMETERS	The function takes the following arguments: <table><tr><td><i>y</i></td><td>Pointer to destination Y component row. <i>y</i> must be 8-byte aligned.</td></tr><tr><td><i>cb</i></td><td>Pointer to destination Cb component row. <i>cb</i> must be 8-byte aligned.</td></tr><tr><td><i>cr</i></td><td>Pointer to destination Cr component row. <i>cr</i> must be 8-byte aligned.</td></tr><tr><td><i>abgr</i></td><td>Pointer to source ABGR multi-component row. <i>abgr</i> must be 8-byte aligned.</td></tr><tr><td><i>n</i></td><td>Length of Y component row. <i>n</i> must be even. The length of Cb and Cr component rows must be <i>n</i>/2. The length of the RGB multi-component row must be 3*<i>n</i>.</td></tr></table>	<i>y</i>	Pointer to destination Y component row. <i>y</i> must be 8-byte aligned.	<i>cb</i>	Pointer to destination Cb component row. <i>cb</i> must be 8-byte aligned.	<i>cr</i>	Pointer to destination Cr component row. <i>cr</i> must be 8-byte aligned.	<i>abgr</i>	Pointer to source ABGR multi-component row. <i>abgr</i> must be 8-byte aligned.	<i>n</i>	Length of Y component row. <i>n</i> must be even. The length of Cb and Cr component rows must be <i>n</i> /2. The length of the RGB multi-component row must be 3* <i>n</i> .
<i>y</i>	Pointer to destination Y component row. <i>y</i> must be 8-byte aligned.										
<i>cb</i>	Pointer to destination Cb component row. <i>cb</i> must be 8-byte aligned.										
<i>cr</i>	Pointer to destination Cr component row. <i>cr</i> must be 8-byte aligned.										
<i>abgr</i>	Pointer to source ABGR multi-component row. <i>abgr</i> must be 8-byte aligned.										
<i>n</i>	Length of Y component row. <i>n</i> must be even. The length of Cb and Cr component rows must be <i>n</i> /2. The length of the RGB multi-component row must be 3* <i>n</i> .										
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .										
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe				
ATTRIBUTE TYPE	ATTRIBUTE VALUE										
Interface Stability	Evolving										
MT-Level	MT-Safe										
SEE ALSO	<code>mllib_VideoColorABGR2JFIFYCC420(3MLIB)</code> , <code>mllib_VideoColorARGB2JFIFYCC420(3MLIB)</code> , <code>mllib_VideoColorARGB2JFIFYCC422(3MLIB)</code> , <code>mllib_VideoColorRGB2JFIFYCC420(3MLIB)</code> , <code>mllib_VideoColorRGB2JFIFYCC422(3MLIB)</code> , <code>attributes(5)</code>										

NAME mllib_VideoColorABGR2JFIFYCC444 – ABGR to JFIF YCbCr color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorABGR2JFIFYCC444(mllib_u8 *y, mllib_u8
    *cb, mllib_u8 *cr, const mllib_u8 *abgr, mllib_s32 n);
```

DESCRIPTION The mllib_VideoColorABGR2JFIFYCC444 () function performs color space conversion from and ABGR to YCbCr when used in the JPEG File Interchange Format (JFIF).

PARAMETERS The function takes the following arguments:

y Pointer to destination Y component row. *y* must be 8-byte aligned.

cb Pointer to destination Cb component row. *cb* must be 8-byte aligned.

cr Pointer to destination Cr component row. *cr* must be 8-byte aligned.

abgr Pointer to source ABGR multi-component row. *abgr* must be 8-byte aligned.

n Length of Y component row. The length of Cb and Cr component rows must be *n*. The length of the RGB multi-component row must be 3**n*.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorARGB2JFIFYCC444(3MLIB),
 mllib_VideoColorRGB2JFIFYCC444(3MLIB),
 mllib_VideoColorRGB2JFIFYCC444_S16(3MLIB), attributes(5)

mllib_VideoColorABGR2RGB(3MLIB)

NAME | mllib_VideoColorABGR2RGB – color conversion

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorABGR2RGB(mllib_u8 *rgb, const mllib_u8
    *abgr, mllib_s32 n);
```

DESCRIPTION | The mllib_VideoColorABGR2RGB() function performs ABGR to RGB color order conversion.

PARAMETERS | The function takes the following arguments:

<i>rgb</i>	Pointer to RGB row.
<i>abgr</i>	Pointer to ABGR row.
<i>n</i>	Number of pixels.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_VideoColorARGB2RGB\(3MLIB\)](#), [mllib_VideoColorRGB2ABGR\(3MLIB\)](#), [mllib_VideoColorRGB2ARGB\(3MLIB\)](#), [attributes\(5\)](#)

mllib_VideoColorABGRint_to_ARGBint(3MLIB)

NAME | mllib_VideoColorABGRint_to_ARGBint – convert ABGR interleaved to ARGB

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_VideoColorABGRint_to_ARGBint(mllib_u32 *ARGB, const
    mllib_u32 *ABGR, mllib_s32 w, mllib_s32 h, mllib_s32 dlb,
    mllib_s32 slb);
```

DESCRIPTION | The ABGR pixel stream is broken apart and recombined into an ARGB pixel stream. All pixel components are 8-bit unsigned integers. The buffers have dimensions *w* and *h*. Within each 32-bit input word, the component ordering is A (bits 31-24), B (bits 23-16), G (bits 15-8), and R (bits 7-0). Within each 32-bit output word, the component ordering is A (bits 31-24), R (bits 23-16), G (bits 15-8), and B (bits 7-0).

PARAMETERS | The function takes the following arguments:

ARGB | Pointer to output buffer.

ABGR | Pointer to input buffer.

w | Image width in pixels.

h | Image height in lines.

dlb | Linebytes for output buffer.

slb | Linebytes for input buffer.

RETURN VALUES | None.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_VideoColorRGBaint_to_ABGRint(3MLIB)`,
`mllib_VideoColorBGRaint_to_ABGRint(3MLIB)`, `attributes(5)`

mllib_VideoColorARGB2JFIFYCC420(3MLIB)

NAME | mllib_VideoColorARGB2JFIFYCC420 – ARGB to JFIF YCbCr color conversion

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorARGB2JFIFYCC420(mllib_u8 *y0, mllib_u8
*y1, mllib_u8 *cb, mllib_u8 *cr, const mllib_u8 *argb0, const
mllib_u8 *argb1, mllib_s32 n);
```

DESCRIPTION | The mllib_VideoColorARGB2JFIFYCC420() function performs color space conversion from ARGB to YCbCr to together with sampling rate conversion when used in the JPEG File Interchange Format (JFIF).

PARAMETERS | The function takes the following arguments:

y0 | Pointer to upper destination Y component row. *y0* must be 8-byte aligned.

y1 | Pointer to lower destination Y component row. *y1* must be 8-byte aligned.

cb | Pointer to destination Cb component row. *cb* must be 8-byte aligned.

cr | Pointer to destination Cr component row. *cr* must be 8-byte aligned.

argb0 | Pointer to upper source ARGB multi-component row. *argb0* must be 8-byte aligned.

argb1 | Pointer to lower source ARGB multi-component row. *argb1* must be 8-byte aligned.

n | Length of Y component row. *n* must be even. The length of Cb and Cr component rows must be *n*/2. The length of the RGB multi-component row must be 3**n*.

RETURN VALUES | The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_VideoColorABGR2JFIFYCC420(3MLIB), mllib_VideoColorABGR2JFIFYCC422(3MLIB), mllib_VideoColorARGB2JFIFYCC422(3MLIB), mllib_VideoColorRGB2JFIFYCC420(3MLIB), mllib_VideoColorRGB2JFIFYCC422(3MLIB), attributes(5)

mllib_VideoColorARGB2JFIFYCC422(3MLIB)

NAME mllib_VideoColorARGB2JFIFYCC422 – ARGB to JFIF YCbCr color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorARGB2JFIFYCC422 (mllib_u8 *y, mllib_u8
    *cb, mllib_u8 *cr, const mllib_u8 *argb, mllib_s32 n);
```

DESCRIPTION The mllib_VideoColorARGB2JFIFYCC422 () function performs color space conversion from ARGB to YCbCr to together with sampling rate conversion when used in the JPEG File Interchange Format (JFIF).

PARAMETERS The function takes the following arguments:

y Pointer to destination Y component row. *y* must be 8-byte aligned.

cb Pointer to destination Cb component row. *cb* must be 8-byte aligned.

cr Pointer to destination Cr component row. *cr* must be 8-byte aligned.

argb Pointer to source ARGB multi-component row. *argb* must be 8-byte aligned.

n Length of Y component row. *n* must be even. The length of Cb and Cr component rows must be *n*/2. The length of the RGB multi-component row must be 3**n*.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorABGR2JFIFYCC420(3MLIB),
 mllib_VideoColorABGR2JFIFYCC422(3MLIB),
 mllib_VideoColorARGB2JFIFYCC420(3MLIB),
 mllib_VideoColorRGB2JFIFYCC420(3MLIB),
 mllib_VideoColorRGB2JFIFYCC422(3MLIB), attributes(5)

mllib_VideoColorARGB2JFIFYCC444(3MLIB)

NAME	mllib_VideoColorARGB2JFIFYCC444 – ARGB to JFIF YCbCr color conversion										
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoColorARGB2JFIFYCC444 (mllib_u8 *y, mllib_u8 *cb, mllib_u8 *cr, const mllib_u8 *argb, mllib_s32 n);</pre>										
DESCRIPTION	The <code>mllib_VideoColorARGB2JFIFYCC444()</code> function performs color space conversion from ARGB to YCbCr when used in the JPEG File Interchange Format (JFIF).										
PARAMETERS	The function takes the following arguments: <table><tr><td><i>y</i></td><td>Pointer to destination Y component row. <i>y</i> must be 8-byte aligned.</td></tr><tr><td><i>cb</i></td><td>Pointer to destination Cb component row. <i>cb</i> must be 8-byte aligned.</td></tr><tr><td><i>cr</i></td><td>Pointer to destination Cr component row. <i>cr</i> must be 8-byte aligned.</td></tr><tr><td><i>argb</i></td><td>Pointer to source ARGB multi-component row. <i>argb</i> must be 8-byte aligned.</td></tr><tr><td><i>n</i></td><td>Length of Y component row. The length of Cb and Cr component rows must be <i>n</i>. The length of the RGB multi-component row must be 3*<i>n</i>.</td></tr></table>	<i>y</i>	Pointer to destination Y component row. <i>y</i> must be 8-byte aligned.	<i>cb</i>	Pointer to destination Cb component row. <i>cb</i> must be 8-byte aligned.	<i>cr</i>	Pointer to destination Cr component row. <i>cr</i> must be 8-byte aligned.	<i>argb</i>	Pointer to source ARGB multi-component row. <i>argb</i> must be 8-byte aligned.	<i>n</i>	Length of Y component row. The length of Cb and Cr component rows must be <i>n</i> . The length of the RGB multi-component row must be 3* <i>n</i> .
<i>y</i>	Pointer to destination Y component row. <i>y</i> must be 8-byte aligned.										
<i>cb</i>	Pointer to destination Cb component row. <i>cb</i> must be 8-byte aligned.										
<i>cr</i>	Pointer to destination Cr component row. <i>cr</i> must be 8-byte aligned.										
<i>argb</i>	Pointer to source ARGB multi-component row. <i>argb</i> must be 8-byte aligned.										
<i>n</i>	Length of Y component row. The length of Cb and Cr component rows must be <i>n</i> . The length of the RGB multi-component row must be 3* <i>n</i> .										
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .										
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe				
ATTRIBUTE TYPE	ATTRIBUTE VALUE										
Interface Stability	Evolving										
MT-Level	MT-Safe										
SEE ALSO	<code>mllib_VideoColorABGR2JFIFYCC444(3MLIB)</code> , <code>mllib_VideoColorRGB2JFIFYCC444(3MLIB)</code> , <code>mllib_VideoColorRGB2JFIFYCC444_S16(3MLIB)</code> , <code>attributes(5)</code>										

NAME mllib_VideoColorARGB2RGB – color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorARGB2RGB(mllib_u8 *rgb, const mllib_u8
    *argb, mllib_s32 n);
```

DESCRIPTION The mllib_VideoColorARGB2RGB() function performs ARGB to RGB color order conversion.

PARAMETERS The function takes the following arguments:

rgb Pointer to RGB row.

argb Pointer to ARGB row.

n Number of pixels.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorABGR2RGB(3MLIB), mllib_VideoColorRGB2ABGR(3MLIB), mllib_VideoColorRGB2ARGB(3MLIB), attributes(5)

mllib_VideoColorBGRaint_to_ABGRint(3MLIB)

NAME mllib_VideoColorBGRaint_to_ABGRint – convert BGRA interleaved to ABGR

SYNOPSIS cc [*flag...*] *file...* -lmllib [*library...*]
#include <mllib.h>

```
void mllib_VideoColorBGRaint_to_ABGRint(mllib_u32 *ABGR, const
    mllib_u32 *BGRA, mllib_s32 w, mllib_s32 h, mllib_s32 dlb,
    mllib_s32 slb);
```

DESCRIPTION The BGRA pixel stream is broken apart and recombined into an ABGR pixel stream. All pixel components are 8-bit unsigned integers. The buffers have dimensions *w* and *h*. Within each 32-bit input word, the component ordering is B (bits 31-24), G (bits 23-16), R (bits 15-8), and A (bits 7-0). Within each 32-bit output word, the component ordering is A (bits 31-24), B (bits 23-16), G (bits 15-8), and R (bits 7-0).

PARAMETERS The function takes the following arguments:

<i>ABGR</i>	Pointer to output buffer.
<i>BGRA</i>	Pointer to input buffer.
<i>w</i>	Image width in pixels.
<i>h</i>	Image height in lines.
<i>dlb</i>	Linebytes for output buffer.
<i>slb</i>	Linebytes for input buffer.

RETURN VALUES None.

ATTRIBUTES See [attributes\(5\)](#) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_VideoColorABGRint_to_ARGBint\(3MLIB\)](#),
[mllib_VideoColorRGBaint_to_ABGRint\(3MLIB\)](#), [attributes\(5\)](#)

NAME mllib_VideoColorBGRint_to_ABGRint – convert BGR interleaved to ABGR interleaved

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_VideoColorBGRint_to_ABGRint(mllib_u32 *ABGR, const
    mllib_u8 *BGR, const mllib_u8 *A_array, mllib_u8 A_const, mllib_s32
    w, mllib_s32 h, mllib_s32 dlb, mllib_s32 slb, mllib_s32 alb);
```

DESCRIPTION

The interleaved BGR stream, and the A values are combined into an A, B, G, R interleaved byte stream. Within each 24-bit input pixel, the component ordering is B (bits 23-16), G (bits 15-8), and R (bits 7-0). Within each 32-bit output word, the component ordering is A (bits 31-24), B (bits 23-16), G (bits 15-8), and R (bits 7-0).

The alpha values for this function work in the following fashion:

- If *A_array* pointer is not NULL, the values are taken from there. It has to have the same dimensions as the R, G, and B buffers.
- If *A_array* pointer is NULL, the alpha values for every pixel are set to *A_const*.

PARAMETERS

The function takes the following arguments:

ABGR Pointer to output buffer.

BGR Pointer to input buffer.

A_array Array of alpha values.

A_const Constant alpha value.

w Image width in pixels.

h Image height in lines.

dlb Linebytes for output buffer.

slb Linebytes for input buffer.

alb Linebytes for alpha buffer.

RETURN VALUES None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

`mlib_VideoColorBGRint_to_ABGRint(3MLIB)`

SEE ALSO | `mlib_VideoColorRGBseq_to_ABGRint(3MLIB)`,
`mlib_VideoColorRGBint_to_ABGRint(3MLIB)`,
`mlib_VideoColorRGBXint_to_ABGRint(3MLIB)`,
`mlib_VideoColorRGBXint_to_ARGBint(3MLIB)`,
`mlib_VideoColorXRGBint_to_ABGRint(3MLIB)`,
`mlib_VideoColorXRGBint_to_ARGBint(3MLIB)`, `attributes(5)`

NAME	mlib_VideoColorBlendABGR, mlib_VideoColorBlendABGR_ResetAlpha – image blend
SYNOPSIS	<pre>cc [flag...] file... -lmlib [library...] #include <mlib.h> void mlib_VideoColorBlendABGR(mlib_u32 *dst, const mlib_u32 *src1, const mlib_u32 *src2, mlib_s32 src1_w, mlib_s32 src1_h, mlib_s32 src2_w, mlib_s32 src2_h, mlib_s32 src2_x, mlib_s32 src2_y, mlib_s32 dst_lb, mlib_s32 src1_lb, mlib_s32 src2_lb, mlib_blend src1_blend, mlib_blend src2_blend); void mlib_VideoColorBlendABGR_ResetAlpha(mlib_u32 *dst, const mlib_u32 *src1, const mlib_u32 *src2, mlib_s32 src1_w, mlib_s32 src1_h, mlib_s32 src2_w, mlib_s32 src2_h, mlib_s32 src2_x, mlib_s32 src2_y, mlib_s32 dst_lb, mlib_s32 src1_lb, mlib_s32 src2_lb, mlib_blend src1_blend, mlib_blend src2_blend);</pre>
DESCRIPTION	<p>The functions use the following equation for blending images:</p> $dst = (src1 * src1_blend) + (src2 * src2_blend)$ <p>The two multi-banded source images (src1 and src2) are blended together and stored in the destination image (dst). The image buffers pointed to by dst, src1, and src2 contain 4-banded ABGR images, 8 bits per component. src1_w and src1_h are the dimensions of the src1 input buffer. src2_w and src2_h are the dimensions of the src2 input buffer. The output buffer must be at least as large as the src1 input buffer. src2_x and src2_y are the offset of the src2 input buffer relative to src1. Where pixels in src2 overlap pixels in src1, the pixels are blended. Pixels in src1 which are outside of src2 are copied into dst. Pixels in the dst image outside of src1 are left unchanged. src1_blend specifies the blend function to be applied to the pixels of src1 image and src2_blend specifies the blend function to be applied to the pixels of src2.</p> <p>Possible blend functions are:</p> <pre>MLIB_BLEND_ZERO MLIB_BLEND_ONE MLIB_BLEND_SRC_ALPHA MLIB_BLEND_ONE_MINUS_SRC_ALPHA MLIB_BLEND_DST_ALPHA MLIB_BLEND_ONE_MINUS_DST_ALPHA</pre> <p>MLIB_BLEND_SRC_ALPHA is the alpha component of image src2 scaled to the range 0.0 to 1.0. MLIB_BLEND_DST_ALPHA is the alpha component of image src1 scaled to the range 0.0 to 1.0. All pixel components are treated as unsigned 8-bit quantities and the output pixel component values are clamped to the range 0 to 255.</p> <p>For the mlib_VideoColorBlendABGR_ResetAlpha() function, the alpha value of every pixel in destination image is set to 0 after blending is complete.</p>
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <p><i>dst</i> Pointer to output image.</p>

`mllib_VideoColorBlendABGR(3MLIB)`

src1 Pointer to 1st input image.
src2 Pointer to 2nd input image.
src1_w *src1* image width in pixels.
src1_h *src1* image height in rows.
src2_w *src2* image width in pixels.
src2_h *src2* image height in rows.
src2_x *src2* horizontal displacement (in pixels), relative to the upper-left corner of *src1*.
src2_y *src2* vertical displacement (in rows), relative to the upper-left corner of *src1*.
dst_lb Linebytes for output image.
src1_lb Linebytes for 1st input image.
src2_lb Linebytes for 2nd input image.
src1_blend Blend function for *src1* image.
src2_blend Blend function for *src2* image.

RETURN VALUES None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_VideoColorBlendABGR_Inp\(3MLIB\)](#), `attributes(5)`

NAME	mlib_VideoColorBlendABGR_Inp, mlib_VideoColorBlendABGR_ResetAlpha_Inp – in-place image blend										
SYNOPSIS	<pre>cc [flag...] file... -lmlib [library...] #include <mlib.h> void mlib_VideoColorBlendABGR_Inp(mlib_u32 *src1dst, const mlib_u32 *src2, mlib_s32 src1dst_w, mlib_s32 src1dst_h, mlib_s32 src2_w, mlib_s32 src2_h, mlib_s32 src2_x, mlib_s32 src2_y, mlib_s32 src1dst_lb, mlib_s32 src2_lb, mlib_blend src1dst_blend, mlib_blend src2_blend) ; void mlib_VideoColorBlendABGR_ResetAlpha_Inp(mlib_u32 *src1dst, const mlib_u32 *src2, mlib_s32 src1dst_w, mlib_s32 src1dst_h, mlib_s32 src2_w, mlib_s32 src2_h, mlib_s32 src2_x, mlib_s32 src2_y, mlib_s32 src1dst_lb, mlib_s32 src2_lb, mlib_blend src1dst_blend, mlib_blend src2_blend) ;</pre>										
DESCRIPTION	<p>The functions use the following equation for blending images:</p> $\text{src1dst} = (\text{src1dst} * \text{src1dst_blend}) + (\text{src2} * \text{src2_blend})$ <p>The two multi-banded source images (src1dst and src2) are blended together and the result is stored in src1dst. src1dst_blend specifies the blend function to be applied to the src1dst image and src2_blend specifies the blend function to be applied to the src2 image. src2_x and src2_y specify position of src2 relative to the upper-left corner of src1dst. src2 is clipped to the boundaries of src1dst, if needed.</p> <p>Possible blend functions are:</p> <pre>MLIB_BLEND_ZERO MLIB_BLEND_ONE MLIB_BLEND_SRC_ALPHA MLIB_BLEND_ONE_MINUS_SRC_ALPHA MLIB_BLEND_DST_ALPHA MLIB_BLEND_ONE_MINUS_DST_ALPHA</pre> <p>MLIB_BLEND_DST_ALPHA is the alpha band of image src1 scaled to the range 0 to 1. MLIB_BLEND_SRC_ALPHA is the alpha band of image src2 scaled to the range 0 to 1. The output pixel bands are clamped to the range 0 to 255.</p> <p>For the mlib_VideoColorBlendABGR_ResetAlpha_Inp() function, the alpha value of every pixel in destination image is set to 0 after blending is complete.</p>										
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <table border="0"> <tr> <td style="padding-right: 20px;"><i>src1dst</i></td> <td>Pointer to 1st input image (also dest. image).</td> </tr> <tr> <td><i>src2</i></td> <td>Pointer to 2nd input image.</td> </tr> <tr> <td><i>src1dst_w</i></td> <td>src1dst image width in pixels.</td> </tr> <tr> <td><i>src1dst_h</i></td> <td>src1dst image height in rows.</td> </tr> <tr> <td><i>src2_w</i></td> <td>src2 image width in pixels.</td> </tr> </table>	<i>src1dst</i>	Pointer to 1st input image (also dest. image).	<i>src2</i>	Pointer to 2nd input image.	<i>src1dst_w</i>	src1dst image width in pixels.	<i>src1dst_h</i>	src1dst image height in rows.	<i>src2_w</i>	src2 image width in pixels.
<i>src1dst</i>	Pointer to 1st input image (also dest. image).										
<i>src2</i>	Pointer to 2nd input image.										
<i>src1dst_w</i>	src1dst image width in pixels.										
<i>src1dst_h</i>	src1dst image height in rows.										
<i>src2_w</i>	src2 image width in pixels.										

mllib_VideoColorBlendABGR_Inp(3MLIB)

<i>src2_h</i>	src2 image height in rows.
<i>src2_x</i>	src2 horizontal displacement (in pixels), relative to the upper-left corner of src1dst.
<i>src2_y</i>	src2 vertical displacement (in rows), relative to the upper-left corner of src1dst.
<i>src1dst_lb</i>	Linebytes for src1dst image.
<i>src2_lb</i>	Linebytes for src2 image.
<i>src1dst_blend</i>	Blend function for src1dst image.
<i>src2_blend</i>	Blend function for src2 image.

RETURN VALUES None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_VideoColorBlendABGR\(3MLIB\)](#), `attributes(5)`

mllib_VideoColorCMYK2JFIFYCCK444(3MLIB)

NAME mllib_VideoColorCMYK2JFIFYCCK444 – CMYK to JFIF YCbCr color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorCMYK2JFIFYCCK444(mllib_u8 *y, mllib_u8
    *cb, mllib_u8 *cr, mllib_u8 *k, const mllib_u8 *cmyk, mllib_s32 n);
```

DESCRIPTION The mllib_VideoColorCMYK2JFIFYCCK444 () function performs color space conversion from CMYK to YCbCr when used in the JPEG File Interchange Format (JFIF).

PARAMETERS The function takes the following arguments:

<i>cb</i>	Pointer to destination Cb component row. <i>cb</i> must be 8-byte aligned.
<i>cr</i>	Pointer to destination Cr component row. <i>cr</i> must be 8-byte aligned.
<i>k</i>	Pointer to destination K component row. <i>k</i> must be 8-byte aligned.
<i>cmyk</i>	Pointer to source CMYK multi-component row. <i>cmyk</i> must be 8-byte aligned.
<i>n</i>	Length of Y,Cb,Cr, and K component rows. The length of the CMYK multi-component row must be <i>n</i> . The length of the RGB multi-component row must be 4* <i>n</i> .

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorJFIFYCCK2CMYK444(3MLIB), attributes(5)

mllib_VideoColorJFIFYCC2ABGR444(3MLIB)

NAME mllib_VideoColorJFIFYCC2ABGR444 – JFIF YCbCr to ABGR color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorJFIFYCC2ABGR444 (mllib_u8 *abgr, const
mllib_u8 *y, const mllib_u8 *cb, const mllib_u8 *cr, mllib_s32 n);
```

DESCRIPTION The mllib_VideoColorJFIFYCC2ABGR444 () function performs color space conversion from YCbCr to ABGR when used in the JPEG File Interchange Format (JFIF).

PARAMETERS The function takes the following arguments:

abgr Pointer to destination ABGR multi-component row. abgr must be 8-byte aligned.

y Pointer to source Y component row. y must be 8-byte aligned.

cb Pointer to source Cb component row. cb must be 8-byte aligned.

cr Pointer to source Cr component row. cr must be 8-byte aligned.

n Length of Y component row. The length of Cb and Cr component rows must be n. The length of the RGB multi-component row must be 3*n.

RETURN VALUES The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorJFIFYCC2ARGB444(3MLIB), mllib_VideoColorJFIFYCC2RGB444(3MLIB), mllib_VideoColorJFIFYCC2RGB444_s16(3MLIB), attributes(5)

mllib_VideoColorJFIFYCC2ARGB444(3MLIB)

NAME mllib_VideoColorJFIFYCC2ARGB444 – JFIF YCbCr to ARGB color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorJFIFYCC2ARGB444(mllib_u8 *argb, const
mllib_u8 *y, const mllib_u8 *cb, const mllib_u8 *cr, mllib_s32 n);
```

DESCRIPTION The mllib_VideoColorJFIFYCC2ARGB444 () function performs color space conversion from YCbCr to ARGB when used in the JPEG File Interchange Format (JFIF).

PARAMETERS The function takes the following arguments:

<i>argb</i>	Pointer to destination ARGB multi-component row. <i>argb</i> must be 8-byte aligned.
<i>y</i>	Pointer to source Y component row. <i>y</i> must be 8-byte aligned.
<i>cb</i>	Pointer to source Cb component row. <i>cb</i> must be 8-byte aligned.
<i>cr</i>	Pointer to source Cr component row. <i>cr</i> must be 8-byte aligned.
<i>n</i>	Length of Y component row. The length of Cb and Cr component rows must be <i>n</i> . The length of the RGB multi-component row must be 3* <i>n</i> .

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorJFIFYCC2ABGR444(3MLIB),
mllib_VideoColorJFIFYCC2RGB444(3MLIB),
mllib_VideoColorJFIFYCC2RGB444_s16(3MLIB), attributes(5)

mllib_VideoColorJFIFYCC2RGB420(3MLIB)

NAME	mllib_VideoColorJFIFYCC2RGB420 – JFIF YCbCr to RGB color conversion																						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_VideoColorJFIFYCC2RGB420(mllib_u8 *<i>rgb0</i>, mllib_u8 *<i>rgb1</i>, const mllib_u8 *<i>y0</i>, const mllib_u8 *<i>y1</i>, const mllib_u8 *<i>cb0</i>, const mllib_u8 *<i>cr0</i>, const mllib_u8 *<i>cb1</i>, const mllib_u8 *<i>cr1</i>, const mllib_u8 *<i>cb2</i>, const mllib_u8 *<i>cr2</i>, mllib_s32 <i>n</i>);</pre>																						
DESCRIPTION	The mllib_VideoColorJFIFYCC2RGB420 () function performs color space conversion from YCbCr to RGB together with sampling rate conversion when used in the JPEG File Interchange Format (JFIF).																						
PARAMETERS	The function takes the following arguments: <table><tr><td><i>rgb0</i></td><td>Pointer to upper destination RGB multi-component row. <i>rgb0</i> must be 8-byte aligned.</td></tr><tr><td><i>rgb1</i></td><td>Pointer to lower destination RGB multi-component row. <i>rgb1</i> must be 8-byte aligned.</td></tr><tr><td><i>y0</i></td><td>Pointer to upper destination Y component row. <i>y0</i> must be 8-byte aligned.</td></tr><tr><td><i>y1</i></td><td>Pointer to lower destination Y component row. <i>y1</i> must be 8-byte aligned.</td></tr><tr><td><i>cb0</i></td><td>Pointer to source upper Cb component row. <i>cb0</i> must be 8-byte aligned.</td></tr><tr><td><i>cr0</i></td><td>Pointer to source upper Cr component row. <i>cr0</i> must be 8-byte aligned.</td></tr><tr><td><i>cb1</i></td><td>Pointer to source middle Cb component row. <i>cb1</i> must be 8-byte aligned.</td></tr><tr><td><i>cr1</i></td><td>Pointer to source middle Cr component row. <i>cr1</i> must be 8-byte aligned.</td></tr><tr><td><i>cb2</i></td><td>Pointer to source lower Cb component row. <i>cb2</i> must be 8-byte aligned.</td></tr><tr><td><i>cr2</i></td><td>Pointer to source lower Cr component row. <i>cr2</i> must be 8-byte aligned.</td></tr><tr><td><i>n</i></td><td>Length of Y component row. The length of Cb and Cr component rows must be <i>n</i>. The length of the RGB multi-component row must be 3*<i>n</i>.</td></tr></table>	<i>rgb0</i>	Pointer to upper destination RGB multi-component row. <i>rgb0</i> must be 8-byte aligned.	<i>rgb1</i>	Pointer to lower destination RGB multi-component row. <i>rgb1</i> must be 8-byte aligned.	<i>y0</i>	Pointer to upper destination Y component row. <i>y0</i> must be 8-byte aligned.	<i>y1</i>	Pointer to lower destination Y component row. <i>y1</i> must be 8-byte aligned.	<i>cb0</i>	Pointer to source upper Cb component row. <i>cb0</i> must be 8-byte aligned.	<i>cr0</i>	Pointer to source upper Cr component row. <i>cr0</i> must be 8-byte aligned.	<i>cb1</i>	Pointer to source middle Cb component row. <i>cb1</i> must be 8-byte aligned.	<i>cr1</i>	Pointer to source middle Cr component row. <i>cr1</i> must be 8-byte aligned.	<i>cb2</i>	Pointer to source lower Cb component row. <i>cb2</i> must be 8-byte aligned.	<i>cr2</i>	Pointer to source lower Cr component row. <i>cr2</i> must be 8-byte aligned.	<i>n</i>	Length of Y component row. The length of Cb and Cr component rows must be <i>n</i> . The length of the RGB multi-component row must be 3* <i>n</i> .
<i>rgb0</i>	Pointer to upper destination RGB multi-component row. <i>rgb0</i> must be 8-byte aligned.																						
<i>rgb1</i>	Pointer to lower destination RGB multi-component row. <i>rgb1</i> must be 8-byte aligned.																						
<i>y0</i>	Pointer to upper destination Y component row. <i>y0</i> must be 8-byte aligned.																						
<i>y1</i>	Pointer to lower destination Y component row. <i>y1</i> must be 8-byte aligned.																						
<i>cb0</i>	Pointer to source upper Cb component row. <i>cb0</i> must be 8-byte aligned.																						
<i>cr0</i>	Pointer to source upper Cr component row. <i>cr0</i> must be 8-byte aligned.																						
<i>cb1</i>	Pointer to source middle Cb component row. <i>cb1</i> must be 8-byte aligned.																						
<i>cr1</i>	Pointer to source middle Cr component row. <i>cr1</i> must be 8-byte aligned.																						
<i>cb2</i>	Pointer to source lower Cb component row. <i>cb2</i> must be 8-byte aligned.																						
<i>cr2</i>	Pointer to source lower Cr component row. <i>cr2</i> must be 8-byte aligned.																						
<i>n</i>	Length of Y component row. The length of Cb and Cr component rows must be <i>n</i> . The length of the RGB multi-component row must be 3* <i>n</i> .																						
RETURN VALUES	The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.																						

`mllib_VideoColorJFIFYCC2RGB420(3MLIB)`

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_VideoColorJFIFYCC2RGB420_Nearest(3MLIB)`,
`mllib_VideoColorJFIFYCC2RGB422(3MLIB)`,
`mllib_VideoColorJFIFYCC2RGB422_Nearest(3MLIB)`, `attributes(5)`

mllib_VideoColorJFIFYCC2RGB420_Nearest(3MLIB)

NAME	mllib_VideoColorJFIFYCC2RGB420_Nearest – JFIF YCbCr to RGB color conversion														
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoColorJFIFYCC2RGB420_Nearest(mllib_u8 *rgb0, mllib_u8 *rgb1, const mllib_u8 *y0, const mllib_u8 *y1, const mllib_u8 *cb, const mllib_u8 *cr, mllib_s32 n);</pre>														
DESCRIPTION	The <code>mllib_VideoColorJFIFYCC2RGB420_Nearest()</code> function performs color space conversion from YCbCr to RGB together with sampling rate conversion when used in the JPEG File Interchange Format (JFIF).														
PARAMETERS	The function takes the following arguments: <table><tr><td><i>rgb0</i></td><td>Pointer to upper destination RGB multi-component row. <i>rgb0</i> must be 8-byte aligned.</td></tr><tr><td><i>rgb1</i></td><td>Pointer to lower destination RGB multi-component row. <i>rgb1</i> must be 8-byte aligned.</td></tr><tr><td><i>y0</i></td><td>Pointer to upper destination Y component row. <i>y0</i> must be 8-byte aligned.</td></tr><tr><td><i>y1</i></td><td>Pointer to lower destination Y component row. <i>y1</i> must be 8-byte aligned.</td></tr><tr><td><i>cb</i></td><td>Pointer to source Cb component row. <i>cb</i> must be 8-byte aligned.</td></tr><tr><td><i>cr</i></td><td>Pointer to source Cr component row. <i>cr</i> must be 8-byte aligned.</td></tr><tr><td><i>n</i></td><td>Length of Y component row. The length of Cb and Cr component rows must be <i>n</i>. The length of the RGB multi-component row must be 3*<i>n</i>.</td></tr></table>	<i>rgb0</i>	Pointer to upper destination RGB multi-component row. <i>rgb0</i> must be 8-byte aligned.	<i>rgb1</i>	Pointer to lower destination RGB multi-component row. <i>rgb1</i> must be 8-byte aligned.	<i>y0</i>	Pointer to upper destination Y component row. <i>y0</i> must be 8-byte aligned.	<i>y1</i>	Pointer to lower destination Y component row. <i>y1</i> must be 8-byte aligned.	<i>cb</i>	Pointer to source Cb component row. <i>cb</i> must be 8-byte aligned.	<i>cr</i>	Pointer to source Cr component row. <i>cr</i> must be 8-byte aligned.	<i>n</i>	Length of Y component row. The length of Cb and Cr component rows must be <i>n</i> . The length of the RGB multi-component row must be 3* <i>n</i> .
<i>rgb0</i>	Pointer to upper destination RGB multi-component row. <i>rgb0</i> must be 8-byte aligned.														
<i>rgb1</i>	Pointer to lower destination RGB multi-component row. <i>rgb1</i> must be 8-byte aligned.														
<i>y0</i>	Pointer to upper destination Y component row. <i>y0</i> must be 8-byte aligned.														
<i>y1</i>	Pointer to lower destination Y component row. <i>y1</i> must be 8-byte aligned.														
<i>cb</i>	Pointer to source Cb component row. <i>cb</i> must be 8-byte aligned.														
<i>cr</i>	Pointer to source Cr component row. <i>cr</i> must be 8-byte aligned.														
<i>n</i>	Length of Y component row. The length of Cb and Cr component rows must be <i>n</i> . The length of the RGB multi-component row must be 3* <i>n</i> .														
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .														
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe								
ATTRIBUTE TYPE	ATTRIBUTE VALUE														
Interface Stability	Evolving														
MT-Level	MT-Safe														
SEE ALSO	<code>mllib_VideoColorJFIFYCC2RGB420(3MLIB)</code> , <code>mllib_VideoColorJFIFYCC2RGB422(3MLIB)</code> , <code>mllib_VideoColorJFIFYCC2RGB422_Nearest(3MLIB)</code> , <code>attributes(5)</code>														

mllib_VideoColorJFIFYCC2RGB422(3MLIB)

NAME mllib_VideoColorJFIFYCC2RGB422 – JFIF YCbCr to RGB color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorJFIFYCC2RGB422(mllib_u8 *rgb, const
      mllib_u8 *y, const mllib_u8 *cb, const mllib_u8 *cr, mllib_s32 n);
```

DESCRIPTION The mllib_VideoColorJFIFYCC2RGB422() function performs color space conversion from YCbCr to RGB together with sampling rate conversion when used in the JPEG File Interchange Format (JFIF).

PARAMETERS The function takes the following arguments:

<i>rgb</i>	Pointer to destination RGB multi-component row. <i>rgb</i> must be 8-byte aligned.
<i>y</i>	Pointer to destination Y component row. <i>y</i> must be 8-byte aligned.
<i>cb</i>	Pointer to source Cb component row. <i>cb</i> must be 8-byte aligned.
<i>cr</i>	Pointer to source Cr component row. <i>cr</i> must be 8-byte aligned.
<i>n</i>	Length of Y component row. The length of Cb and Cr component rows must be <i>n</i> . The length of the RGB multi-component row must be 3* <i>n</i> .

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorJFIFYCC2RGB420(3MLIB),
mllib_VideoColorJFIFYCC2RGB420_Nearest(3MLIB),
mllib_VideoColorJFIFYCC2RGB422_Nearest(3MLIB), attributes(5)

mllib_VideoColorJFIFYCC2RGB422_Nearest(3MLIB)

NAME	mllib_VideoColorJFIFYCC2RGB422_Nearest – JFIF YCbCr to RGB color conversion										
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoColorJFIFYCC2RGB422_Nearest(mllib_u8 *rgb, const mllib_u8 *y, const mllib_u8 *cb, const mllib_u8 *cr, mllib_s32 n);</pre>										
DESCRIPTION	The <code>mllib_VideoColorJFIFYCC2RGB422_Nearest()</code> function performs color space conversion from YCbCr to RGB together with sampling rate conversion when used in the JPEG File Interchange Format (JFIF).										
PARAMETERS	The function takes the following arguments: <table><tr><td><i>rgb</i></td><td>Pointer to destination RGB multi-component row. <i>rgb</i> must be 8-byte aligned.</td></tr><tr><td><i>y</i></td><td>Pointer to destination Y component row. <i>y</i> must be 8-byte aligned.</td></tr><tr><td><i>cb</i></td><td>Pointer to source Cb component row. <i>cb</i> must be 8-byte aligned.</td></tr><tr><td><i>cr</i></td><td>Pointer to source Cr component row. <i>cr</i> must be 8-byte aligned.</td></tr><tr><td><i>n</i></td><td>Length of Y component row. The length of Cb and Cr component rows must be <i>n</i>. The length of the RGB multi-component row must be 3*<i>n</i>.</td></tr></table>	<i>rgb</i>	Pointer to destination RGB multi-component row. <i>rgb</i> must be 8-byte aligned.	<i>y</i>	Pointer to destination Y component row. <i>y</i> must be 8-byte aligned.	<i>cb</i>	Pointer to source Cb component row. <i>cb</i> must be 8-byte aligned.	<i>cr</i>	Pointer to source Cr component row. <i>cr</i> must be 8-byte aligned.	<i>n</i>	Length of Y component row. The length of Cb and Cr component rows must be <i>n</i> . The length of the RGB multi-component row must be 3* <i>n</i> .
<i>rgb</i>	Pointer to destination RGB multi-component row. <i>rgb</i> must be 8-byte aligned.										
<i>y</i>	Pointer to destination Y component row. <i>y</i> must be 8-byte aligned.										
<i>cb</i>	Pointer to source Cb component row. <i>cb</i> must be 8-byte aligned.										
<i>cr</i>	Pointer to source Cr component row. <i>cr</i> must be 8-byte aligned.										
<i>n</i>	Length of Y component row. The length of Cb and Cr component rows must be <i>n</i> . The length of the RGB multi-component row must be 3* <i>n</i> .										
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .										
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe				
ATTRIBUTE TYPE	ATTRIBUTE VALUE										
Interface Stability	Evolving										
MT-Level	MT-Safe										
SEE ALSO	<code>mllib_VideoColorJFIFYCC2RGB420(3MLIB)</code> , <code>mllib_VideoColorJFIFYCC2RGB420_Nearest(3MLIB)</code> , <code>mllib_VideoColorJFIFYCC2RGB422(3MLIB)</code> , <code>attributes(5)</code>										

mllib_VideoColorJFIFYCC2RGB444(3MLIB)

NAME mllib_VideoColorJFIFYCC2RGB444 – JFIF YCbCr to RGB color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorJFIFYCC2RGB444(mllib_u8 *rgb, const
      mllib_u8 *y, const mllib_u8 *cb, const mllib_u8 *cr, mllib_s32 n);
```

DESCRIPTION The mllib_VideoColorJFIFYCC2RGB444 () function performs color space conversion from YCbCr to RGB when used in the JPEG File Interchange Format (JFIF).

PARAMETERS The function takes the following arguments:

<i>rgb</i>	Pointer to destination RGB multi-component row. <i>rgb</i> must be 8-byte aligned.
<i>y</i>	Pointer to source Y component row. <i>y</i> must be 8-byte aligned.
<i>cb</i>	Pointer to source Cb component row. <i>cb</i> must be 8-byte aligned.
<i>cr</i>	Pointer to source Cr component row. <i>cr</i> must be 8-byte aligned.
<i>n</i>	Length of Y component row. The length of Cb and Cr component rows must be <i>n</i> . The length of the RGB multi-component row must be 3* <i>n</i> .

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorJFIFYCC2ABGR444(3MLIB),
mllib_VideoColorJFIFYCC2ARGB444(3MLIB),
mllib_VideoColorJFIFYCC2RGB444_s16(3MLIB), attributes(5)

mllib_VideoColorJFIFYCC2RGB444_S16(3MLIB)

NAME	mllib_VideoColorJFIFYCC2RGB444_S16 – JFIF YCbCr to RGB color conversion										
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoColorJFIFYCC2RGB444_S16(mllib_s16 *rgb, const mllib_s16 *y, const mllib_s16 *cb, const mllib_s16 *cr, mllib_s32 n);</pre>										
DESCRIPTION	The <code>mllib_VideoColorJFIFYCC2RGB444_S16()</code> function performs color space conversion from YCbCr to RGB when used in the JPEG File Interchange Format (JFIF).										
PARAMETERS	The function takes the following arguments: <table><tr><td><i>rgb</i></td><td>Pointer to destination RGB multi-component row. <i>rgb</i> must be 8-byte aligned.</td></tr><tr><td><i>y</i></td><td>Pointer to source Y component row. <i>y</i> must be 8-byte aligned.</td></tr><tr><td><i>cb</i></td><td>Pointer to source Cb component row. <i>cb</i> must be 8-byte aligned.</td></tr><tr><td><i>cr</i></td><td>Pointer to source Cr component row. <i>cr</i> must be 8-byte aligned.</td></tr><tr><td><i>n</i></td><td>Length of Y component row. The length of Cb and Cr component rows must be <i>n</i>. The length of the RGB multi-component row must be 3*n.</td></tr></table>	<i>rgb</i>	Pointer to destination RGB multi-component row. <i>rgb</i> must be 8-byte aligned.	<i>y</i>	Pointer to source Y component row. <i>y</i> must be 8-byte aligned.	<i>cb</i>	Pointer to source Cb component row. <i>cb</i> must be 8-byte aligned.	<i>cr</i>	Pointer to source Cr component row. <i>cr</i> must be 8-byte aligned.	<i>n</i>	Length of Y component row. The length of Cb and Cr component rows must be <i>n</i> . The length of the RGB multi-component row must be 3*n.
<i>rgb</i>	Pointer to destination RGB multi-component row. <i>rgb</i> must be 8-byte aligned.										
<i>y</i>	Pointer to source Y component row. <i>y</i> must be 8-byte aligned.										
<i>cb</i>	Pointer to source Cb component row. <i>cb</i> must be 8-byte aligned.										
<i>cr</i>	Pointer to source Cr component row. <i>cr</i> must be 8-byte aligned.										
<i>n</i>	Length of Y component row. The length of Cb and Cr component rows must be <i>n</i> . The length of the RGB multi-component row must be 3*n.										
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .										
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe				
ATTRIBUTE TYPE	ATTRIBUTE VALUE										
Interface Stability	Evolving										
MT-Level	MT-Safe										
SEE ALSO	<code>mllib_VideoColorJFIFYCC2ABGR444(3MLIB)</code> , <code>mllib_VideoColorJFIFYCC2ARGB444(3MLIB)</code> , <code>mllib_VideoColorJFIFYCC2RGB444(3MLIB)</code> , <code>attributes(5)</code>										

mllib_VideoColorJFIFYCCK2CMYK444(3MLIB)

NAME mllib_VideoColorJFIFYCCK2CMYK444 – JFIF YCbCr to CMYK color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorJFIFYCCK2CMYK444(mllib_u8 *cmyk, const
mllib_u8 *y, const mllib_u8 *cb, const mllib_u8 *cr, const mllib_u8
*k, mllib_s32 n);
```

DESCRIPTION The `mllib_VideoColorJFIFYCCK2CMYK444()` function performs color space conversion from YCbCr to CMYK when used in the JPEG File Interchange Format (JFIF).

PARAMETERS The function takes the following arguments:

<i>cmyk</i>	Pointer to destination CMYK multi-component row. <i>cmyk</i> must be 8-byte aligned.
<i>y</i>	Pointer to source Y component row. <i>y</i> must be 8-byte aligned.
<i>cb</i>	Pointer to source Cb component row. <i>cb</i> must be 8-byte aligned.
<i>cr</i>	Pointer to source Cr component row. <i>cr</i> must be 8-byte aligned.
<i>k</i>	Pointer to source K component row. <i>k</i> must be 8-byte aligned.
<i>n</i>	Length of Y,Cb,Cr, and K component rows. The length of the CMYK multi-component row must be <i>n</i> . The length of the RGB multi-component row must be 4* <i>n</i> .

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_VideoColorCMYK2JFIFYCCK444\(3MLIB\)](#), [attributes\(5\)](#)

mllib_VideoColorMerge2(3MLIB)

NAME	mllib_VideoColorMerge2 – color conversion (color channel merge)								
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_VideoColorMerge2(mllib_u8 *<i>colors</i>, const mllib_u8 *<i>color1</i>, const mllib_u8 *<i>color2</i>, mllib_s32 <i>n</i>);</pre>								
DESCRIPTION	The mllib_VideoColorMerge2() function performs color channel merge.								
PARAMETERS	The function takes the following arguments: <table><tr><td><i>colors</i></td><td>Pointer to colors multi-component row. colors must be 8-byte aligned.</td></tr><tr><td><i>color1</i></td><td>Pointer to first color component row. color1 must be 8-byte aligned.</td></tr><tr><td><i>color2</i></td><td>Pointer to second color component row. color2 must be 8-byte aligned.</td></tr><tr><td><i>n</i></td><td>Length of color1 and color2 arrays. Length of colors must be 2*n.</td></tr></table>	<i>colors</i>	Pointer to colors multi-component row. colors must be 8-byte aligned.	<i>color1</i>	Pointer to first color component row. color1 must be 8-byte aligned.	<i>color2</i>	Pointer to second color component row. color2 must be 8-byte aligned.	<i>n</i>	Length of color1 and color2 arrays. Length of colors must be 2*n.
<i>colors</i>	Pointer to colors multi-component row. colors must be 8-byte aligned.								
<i>color1</i>	Pointer to first color component row. color1 must be 8-byte aligned.								
<i>color2</i>	Pointer to second color component row. color2 must be 8-byte aligned.								
<i>n</i>	Length of color1 and color2 arrays. Length of colors must be 2*n.								
RETURN VALUES	The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.								
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe		
ATTRIBUTE TYPE	ATTRIBUTE VALUE								
Interface Stability	Evolving								
MT-Level	MT-Safe								
SEE ALSO	mllib_VideoColorMerge2_S16(3MLIB), mllib_VideoColorMerge3(3MLIB), mllib_VideoColorMerge3_S16(3MLIB), mllib_VideoColorMerge4(3MLIB), mllib_VideoColorMerge4_S16(3MLIB), mllib_VideoColorSplit2(3MLIB), mllib_VideoColorSplit2_S16(3MLIB), mllib_VideoColorSplit3(3MLIB), mllib_VideoColorSplit3_S16(3MLIB), mllib_VideoColorSplit4(3MLIB), mllib_VideoColorSplit4_S16(3MLIB), attributes(5)								

NAME | mllib_VideoColorMerge2_S16 – color conversion (color channel merge)

SYNOPSIS | `cc [flag...] file... -lmllib [library...]`
`#include <mllib.h>`

`mllib_status mllib_VideoColorMerge2_S16(mllib_s16 *colors, const
mllib_s16 *color1, const mllib_s16 *color2, mllib_s32 n);`

DESCRIPTION | The mllib_VideoColorMerge2_S16() function performs color channel merge.

PARAMETERS | The function takes the following arguments:

colors | Pointer to colors multi-component row. colors must be 8-byte aligned.

color1 | Pointer to first color component row. color1 must be 8-byte aligned.

color2 | Pointer to second color component row. color2 must be 8-byte aligned.

n | Length of color1 and color2 arrays. Length of colors must be 2*n.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_VideoColorMerge2(3MLIB), mllib_VideoColorMerge3(3MLIB), mllib_VideoColorMerge3_S16(3MLIB), mllib_VideoColorMerge4(3MLIB), mllib_VideoColorMerge4_S16(3MLIB), mllib_VideoColorSplit2(3MLIB), mllib_VideoColorSplit2_S16(3MLIB), mllib_VideoColorSplit3(3MLIB), mllib_VideoColorSplit3_S16(3MLIB), mllib_VideoColorSplit4(3MLIB), mllib_VideoColorSplit4_S16(3MLIB), attributes(5)

mllib_VideoColorMerge3(3MLIB)

NAME	mllib_VideoColorMerge3 – color conversion (color channel merge)										
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_VideoColorMerge3(mllib_u8 *colors, const mllib_u8 *color1, const mllib_u8 *color2, const mllib_u8 *color3, mllib_s32 n);</pre>										
DESCRIPTION	The mllib_VideoColorMerge3() function performs color channel merge.										
PARAMETERS	The function takes the following arguments: <table><tr><td><i>colors</i></td><td>Pointer to colors multi-component row. colors must be 8-byte aligned.</td></tr><tr><td><i>color1</i></td><td>Pointer to first color component row. color1 must be 8-byte aligned.</td></tr><tr><td><i>color2</i></td><td>Pointer to second color component row. color2 must be 8-byte aligned.</td></tr><tr><td><i>color3</i></td><td>Pointer to third color component row. color3 must be 8-byte aligned.</td></tr><tr><td><i>n</i></td><td>Length of color1, color2 and color3 arrays. Length of colors must be 3*n.</td></tr></table>	<i>colors</i>	Pointer to colors multi-component row. colors must be 8-byte aligned.	<i>color1</i>	Pointer to first color component row. color1 must be 8-byte aligned.	<i>color2</i>	Pointer to second color component row. color2 must be 8-byte aligned.	<i>color3</i>	Pointer to third color component row. color3 must be 8-byte aligned.	<i>n</i>	Length of color1, color2 and color3 arrays. Length of colors must be 3*n.
<i>colors</i>	Pointer to colors multi-component row. colors must be 8-byte aligned.										
<i>color1</i>	Pointer to first color component row. color1 must be 8-byte aligned.										
<i>color2</i>	Pointer to second color component row. color2 must be 8-byte aligned.										
<i>color3</i>	Pointer to third color component row. color3 must be 8-byte aligned.										
<i>n</i>	Length of color1, color2 and color3 arrays. Length of colors must be 3*n.										
RETURN VALUES	The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.										
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe				
ATTRIBUTE TYPE	ATTRIBUTE VALUE										
Interface Stability	Evolving										
MT-Level	MT-Safe										
SEE ALSO	mllib_VideoColorMerge2(3MLIB), mllib_VideoColorMerge2_S16(3MLIB), mllib_VideoColorMerge3_S16(3MLIB), mllib_VideoColorMerge4(3MLIB), mllib_VideoColorMerge4_S16(3MLIB), mllib_VideoColorSplit2(3MLIB), mllib_VideoColorSplit2_S16(3MLIB), mllib_VideoColorSplit3(3MLIB), mllib_VideoColorSplit3_S16(3MLIB), mllib_VideoColorSplit4(3MLIB), mllib_VideoColorSplit4_S16(3MLIB), attributes(5)										

NAME mllib_VideoColorMerge3_S16 – color conversion (color channel merge)

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorMerge3_S16(mllib_s16 *colors, const
    mllib_s16 *color1, const mllib_s16 *color2, const mllib_s16 *color3,
    mllib_s32 n);
```

DESCRIPTION The mllib_VideoColorMerge3_S16() function performs color channel merge.

PARAMETERS The function takes the following arguments:

colors Pointer to colors multi-component row. colors must be 8-byte aligned.

color1 Pointer to first color component row. color1 must be 8-byte aligned.

color2 Pointer to second color component row. color2 must be 8-byte aligned.

color3 Pointer to third color component row. color3 must be 8-byte aligned.

n Length of color1, color2 and color3 arrays. Length of colors must be 3*n.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorMerge2(3MLIB), mllib_VideoColorMerge2_S16(3MLIB), mllib_VideoColorMerge3(3MLIB), mllib_VideoColorMerge4(3MLIB), mllib_VideoColorMerge4_S16(3MLIB), mllib_VideoColorSplit2(3MLIB), mllib_VideoColorSplit2_S16(3MLIB), mllib_VideoColorSplit3(3MLIB), mllib_VideoColorSplit3_S16(3MLIB), mllib_VideoColorSplit4(3MLIB), mllib_VideoColorSplit4_S16(3MLIB), attributes(5)

mllib_VideoColorMerge4(3MLIB)

NAME	mllib_VideoColorMerge4 – color conversion (color channel merge)												
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_VideoColorMerge4(mllib_u8 *<i>colors</i>, const mllib_u8 *<i>color1</i>, const mllib_u8 *<i>color2</i>, const mllib_u8 *<i>color3</i>, const mllib_u8 *<i>color4</i>, mllib_s32 <i>n</i>);</pre>												
DESCRIPTION	The mllib_VideoColorMerge4() function performs color channel merge.												
PARAMETERS	The function takes the following arguments: <table><tr><td><i>colors</i></td><td>Pointer to colors multi-component row. colors must be 8-byte aligned.</td></tr><tr><td><i>color1</i></td><td>Pointer to first color component row. color1 must be 8-byte aligned.</td></tr><tr><td><i>color2</i></td><td>Pointer to second color component row. color2 must be 8-byte aligned.</td></tr><tr><td><i>color3</i></td><td>Pointer to third color component row. color3 must be 8-byte aligned.</td></tr><tr><td><i>color4</i></td><td>Pointer to fourth color component row. color4 must be 8-byte aligned.</td></tr><tr><td><i>n</i></td><td>Length of color1, color2, color3, and color4 arrays. Length of colors must be 4*n.</td></tr></table>	<i>colors</i>	Pointer to colors multi-component row. colors must be 8-byte aligned.	<i>color1</i>	Pointer to first color component row. color1 must be 8-byte aligned.	<i>color2</i>	Pointer to second color component row. color2 must be 8-byte aligned.	<i>color3</i>	Pointer to third color component row. color3 must be 8-byte aligned.	<i>color4</i>	Pointer to fourth color component row. color4 must be 8-byte aligned.	<i>n</i>	Length of color1, color2, color3, and color4 arrays. Length of colors must be 4*n.
<i>colors</i>	Pointer to colors multi-component row. colors must be 8-byte aligned.												
<i>color1</i>	Pointer to first color component row. color1 must be 8-byte aligned.												
<i>color2</i>	Pointer to second color component row. color2 must be 8-byte aligned.												
<i>color3</i>	Pointer to third color component row. color3 must be 8-byte aligned.												
<i>color4</i>	Pointer to fourth color component row. color4 must be 8-byte aligned.												
<i>n</i>	Length of color1, color2, color3, and color4 arrays. Length of colors must be 4*n.												
RETURN VALUES	The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.												
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe						
ATTRIBUTE TYPE	ATTRIBUTE VALUE												
Interface Stability	Evolving												
MT-Level	MT-Safe												
SEE ALSO	mllib_VideoColorMerge2(3MLIB), mllib_VideoColorMerge2_S16(3MLIB), mllib_VideoColorMerge3(3MLIB), mllib_VideoColorMerge3_S16(3MLIB), mllib_VideoColorMerge4_S16(3MLIB), mllib_VideoColorSplit2(3MLIB), mllib_VideoColorSplit2_S16(3MLIB), mllib_VideoColorSplit3(3MLIB), mllib_VideoColorSplit3_S16(3MLIB), mllib_VideoColorSplit4(3MLIB), mllib_VideoColorSplit4_S16(3MLIB), attributes(5)												

NAME mllib_VideoColorMerge4_S16 – color conversion (color channel merge)

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorMerge4_S16(mllib_s16 *colors, const
    mllib_s16 *color1, const mllib_s16 *color2, const mllib_s16 *color3,
    const mllib_s16 *color4, mllib_s32 n);
```

DESCRIPTION The mllib_VideoColorMerge4_S16() function performs color channel merge.

PARAMETERS The function takes the following arguments:

colors Pointer to colors multi-component row. colors must be 8-byte aligned.

color1 Pointer to first color component row. color1 must be 8-byte aligned.

color2 Pointer to second color component row. color2 must be 8-byte aligned.

color3 Pointer to third color component row. color3 must be 8-byte aligned.

color4 Pointer to fourth color component row. color4 must be 8-byte aligned.

n Length of color1, color2, color3, and color4 arrays. Length of colors must be 4*n.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorMerge2(3MLIB), mllib_VideoColorMerge2_S16(3MLIB), mllib_VideoColorMerge3(3MLIB), mllib_VideoColorMerge3_S16(3MLIB), mllib_VideoColorMerge4(3MLIB), mllib_VideoColorSplit2(3MLIB), mllib_VideoColorSplit2_S16(3MLIB), mllib_VideoColorSplit3(3MLIB), mllib_VideoColorSplit3_S16(3MLIB), mllib_VideoColorSplit4(3MLIB), mllib_VideoColorSplit4_S16(3MLIB), attributes(5)

mllib_VideoColorResizeABGR(3MLIB)

NAME	mllib_VideoColorResizeABGR – image resize																		
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> void mllib_VideoColorResizeABGR(mllib_u32 *<i>dst</i>, const mllib_u32 *<i>src</i>, mllib_s32 <i>dst_w</i>, mllib_s32 <i>dst_h</i>, mllib_s32 <i>dst_lb</i>, mllib_s32 <i>src_w</i>, mllib_s32 <i>src_h</i>, mllib_s32 <i>src_lb</i>, mllib_filter <i>filter</i>);</pre>																		
DESCRIPTION	The <code>mllib_VideoColorResizeABGR()</code> function resizes the source image with dimensions <code>src_w</code> , <code>src_h</code> into the destination image with dimensions <code>dst_w</code> , <code>dst_h</code> using nearest-neighbor, bilinear interpolation, or bicubic interpolation. The source buffer can contain multi-banded pixel stream, in which case, each band is resized independently. Edge conditions are handled according to the <code>MLIB_EDGE_SRC_EXTEND</code> scheme.																		
PARAMETERS	The function takes the following arguments: <table><tr><td><i>dst</i></td><td>Pointer to output image.</td></tr><tr><td><i>src</i></td><td>Pointer to input image.</td></tr><tr><td><i>dst_w</i></td><td>Output image width in pixels.</td></tr><tr><td><i>dst_h</i></td><td>Output image height in rows.</td></tr><tr><td><i>dst_lb</i></td><td>Input image width in pixels.</td></tr><tr><td><i>src_w</i></td><td>Linebytes for input buffer.</td></tr><tr><td><i>src_h</i></td><td>Input image height in lines.</td></tr><tr><td><i>src_lb</i></td><td>Linebytes for input image.</td></tr><tr><td><i>filter</i></td><td>Type of interpolation filter. It can be one of the following: <code>MLIB_NEAREST</code> <code>MLIB_BILINEAR</code> <code>MLIB_BICUBIC</code></td></tr></table>	<i>dst</i>	Pointer to output image.	<i>src</i>	Pointer to input image.	<i>dst_w</i>	Output image width in pixels.	<i>dst_h</i>	Output image height in rows.	<i>dst_lb</i>	Input image width in pixels.	<i>src_w</i>	Linebytes for input buffer.	<i>src_h</i>	Input image height in lines.	<i>src_lb</i>	Linebytes for input image.	<i>filter</i>	Type of interpolation filter. It can be one of the following: <code>MLIB_NEAREST</code> <code>MLIB_BILINEAR</code> <code>MLIB_BICUBIC</code>
<i>dst</i>	Pointer to output image.																		
<i>src</i>	Pointer to input image.																		
<i>dst_w</i>	Output image width in pixels.																		
<i>dst_h</i>	Output image height in rows.																		
<i>dst_lb</i>	Input image width in pixels.																		
<i>src_w</i>	Linebytes for input buffer.																		
<i>src_h</i>	Input image height in lines.																		
<i>src_lb</i>	Linebytes for input image.																		
<i>filter</i>	Type of interpolation filter. It can be one of the following: <code>MLIB_NEAREST</code> <code>MLIB_BILINEAR</code> <code>MLIB_BICUBIC</code>																		
RETURN VALUES	None.																		
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe												
ATTRIBUTE TYPE	ATTRIBUTE VALUE																		
Interface Stability	Evolving																		
MT-Level	MT-Safe																		
SEE ALSO	<code>attributes(5)</code>																		

NAME mllib_VideoColorRGB2ABGR – color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorRGB2ABGR(mllib_u8 *abgr, const mllib_u8
    *rgb, mllib_s32 n);
```

DESCRIPTION The mllib_VideoColorRGB2ABGR() function performs RGB to ABGR color order conversion.

PARAMETERS The function takes the following arguments:

abgr Pointer to ABGR row.

rgb Pointer to RGB row.

n Number of pixels.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorABGR2RGB(3MLIB), mllib_VideoColorARGB2RGB(3MLIB), mllib_VideoColorRGB2ARGB(3MLIB), attributes(5)

mllib_VideoColorRGB2ARGB(3MLIB)

NAME | mllib_VideoColorRGB2ARGB – color conversion

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorRGB2ARGB(mllib_u8 *argb, const mllib_u8
    *rgb, mllib_s32 n);
```

DESCRIPTION | The mllib_VideoColorRGB2ARGB() function performs RGB to ARGB color order conversion.

PARAMETERS | The function takes the following arguments:

<i>argb</i>	Pointer to ARGB row.
<i>rgb</i>	Pointer to RGB row.
<i>n</i>	Number of pixels.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_VideoColorABGR2RGB\(3MLIB\)](#), [mllib_VideoColorARGB2RGB\(3MLIB\)](#), [mllib_VideoColorRGB2ABGR\(3MLIB\)](#), [attributes\(5\)](#)

NAME mllib_VideoColorRGB2JFIFYCC420 – RGB to JFIF YCbCr color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorRGB2JFIFYCC420(mllib_u8 *y0, mllib_u8
    *y1, mllib_u8 *cb, mllib_u8 *cr, const mllib_u8 *rgb0, const
    mllib_u8 *rgb1, mllib_s32 n);
```

DESCRIPTION The mllib_VideoColorRGB2JFIFYCC420 () function performs color space conversion from RGB to YCbCr to together with sampling rate conversion when used in the JPEG File Interchange Format (JFIF).

PARAMETERS The function takes the following arguments:

y0 Pointer to upper destination Y component row. *y0* must be 8-byte aligned.

y1 Pointer to lower destination Y component row. *y1* must be 8-byte aligned.

cb Pointer to destination Cb component row. *cb* must be 8-byte aligned.

cr Pointer to destination Cr component row. *cr* must be 8-byte aligned.

rgb0 Pointer to upper source RGB multi-component row. *rgb0* must be 8-byte aligned.

rgb1 Pointer to lower source RGB multi-component row. *rgb1* must be 8-byte aligned.

n Length of Y component row. *n* must be even. The length of Cb and Cr component rows must be *n*/2. The length of the RGB multi-component row must be 3**n*.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorABGR2JFIFYCC420(3MLIB), mllib_VideoColorABGR2JFIFYCC422(3MLIB), mllib_VideoColorARGB2JFIFYCC420(3MLIB), mllib_VideoColorARGB2JFIFYCC422(3MLIB), mllib_VideoColorRGB2JFIFYCC422(3MLIB), attributes(5)

mllib_VideoColorRGB2JFIFYCC422(3MLIB)

NAME	mllib_VideoColorRGB2JFIFYCC422 – RGB to JFIF YCbCr color conversion										
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_VideoColorRGB2JFIFYCC422(mllib_u8 *y, mllib_u8 *cb, mllib_u8 *cr, const mllib_u8 *rgb, mllib_s32 n);</pre>										
DESCRIPTION	The <code>mllib_VideoColorRGB2JFIFYCC422()</code> function performs color space conversion from RGB to YCbCr together with sampling rate conversion when used in the JPEG File Interchange Format (JFIF).										
PARAMETERS	The function takes the following arguments: <table><tr><td><i>y</i></td><td>Pointer to destination Y component row. <i>y</i> must be 8-byte aligned.</td></tr><tr><td><i>cb</i></td><td>Pointer to destination Cb component row. <i>cb</i> must be 8-byte aligned.</td></tr><tr><td><i>cr</i></td><td>Pointer to destination Cr component row. <i>cr</i> must be 8-byte aligned.</td></tr><tr><td><i>rgb</i></td><td>Pointer to source RGB multi-component row. <i>rgb</i> must be 8-byte aligned.</td></tr><tr><td><i>n</i></td><td>Length of Y component row. <i>n</i> must be even. The length of Cb and Cr component rows must be <i>n</i>/2. The length of the RGB multi-component row must be 3*<i>n</i>.</td></tr></table>	<i>y</i>	Pointer to destination Y component row. <i>y</i> must be 8-byte aligned.	<i>cb</i>	Pointer to destination Cb component row. <i>cb</i> must be 8-byte aligned.	<i>cr</i>	Pointer to destination Cr component row. <i>cr</i> must be 8-byte aligned.	<i>rgb</i>	Pointer to source RGB multi-component row. <i>rgb</i> must be 8-byte aligned.	<i>n</i>	Length of Y component row. <i>n</i> must be even. The length of Cb and Cr component rows must be <i>n</i> /2. The length of the RGB multi-component row must be 3* <i>n</i> .
<i>y</i>	Pointer to destination Y component row. <i>y</i> must be 8-byte aligned.										
<i>cb</i>	Pointer to destination Cb component row. <i>cb</i> must be 8-byte aligned.										
<i>cr</i>	Pointer to destination Cr component row. <i>cr</i> must be 8-byte aligned.										
<i>rgb</i>	Pointer to source RGB multi-component row. <i>rgb</i> must be 8-byte aligned.										
<i>n</i>	Length of Y component row. <i>n</i> must be even. The length of Cb and Cr component rows must be <i>n</i> /2. The length of the RGB multi-component row must be 3* <i>n</i> .										
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .										
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe				
ATTRIBUTE TYPE	ATTRIBUTE VALUE										
Interface Stability	Evolving										
MT-Level	MT-Safe										
SEE ALSO	<code>mllib_VideoColorABGR2JFIFYCC420(3MLIB)</code> , <code>mllib_VideoColorABGR2JFIFYCC422(3MLIB)</code> , <code>mllib_VideoColorARGB2JFIFYCC420(3MLIB)</code> , <code>mllib_VideoColorARGB2JFIFYCC422(3MLIB)</code> , <code>mllib_VideoColorRGB2JFIFYCC420(3MLIB)</code> , <code>attributes(5)</code>										

mllib_VideoColorRGB2JFIFYCC444(3MLIB)

NAME mllib_VideoColorRGB2JFIFYCC444 – RGB to JFIF YCbCr color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorRGB2JFIFYCC444(mllib_u8 *y, mllib_u8
      *cb, mllib_u8 *cr, const mllib_u8 *rgb, mllib_s32 n);
```

DESCRIPTION The mllib_VideoColorRGB2JFIFYCC444 () function performs color space conversion from RGB to YCbCr when used in the JPEG File Interchange Format (JFIF).

PARAMETERS The function takes the following arguments:

y Pointer to destination Y component row. *y* must be 8-byte aligned.

cb Pointer to destination Cb component row. *cb* must be 8-byte aligned.

cr Pointer to destination Cr component row. *cr* must be 8-byte aligned.

rgb Pointer to source RGB multi-component row. *rgb* must be 8-byte aligned.

n Length of Y component row. The length of Cb and Cr component rows must be *n*. The length of the RGB multi-component row must be 3**n*.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorABGR2JFIFYCC444(3MLIB),
 mllib_VideoColorARGB2JFIFYCC444(3MLIB),
 mllib_VideoColorRGB2JFIFYCC444_S16(3MLIB), attributes(5)

mllib_VideoColorRGB2JFIFYCC444_S16(3MLIB)

NAME	mllib_VideoColorRGB2JFIFYCC444_S16 – RGB to JFIF YCbCr color conversion						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoColorRGB2JFIFYCC444_S16(mllib_s16 *y, mllib_s16 *cb, mllib_s16 *cr, const mllib_s16 *rgb, mllib_s32 n);</pre>						
DESCRIPTION	The <code>mllib_VideoColorRGB2JFIFYCC444_S16()</code> function performs color space conversion from RGB to YCbCr when used in the JPEG File Interchange Format (JFIF).						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>y</i> Pointer to destination Y component row. <i>y</i> must be 8-byte aligned.</p> <p><i>cb</i> Pointer to destination Cb component row. <i>cb</i> must be 8-byte aligned.</p> <p><i>cr</i> Pointer to destination Cr component row. <i>cr</i> must be 8-byte aligned.</p> <p><i>rgb</i> Pointer to source RGB multi-component row. <i>rgb</i> must be 8-byte aligned.</p> <p><i>n</i> Length of Y component row. The length of Cb and Cr component rows must be <i>n</i>. The length of the RGB multi-component row must be 3*<i>n</i>.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_VideoColorABGR2JFIFYCC444(3MLIB)</code> , <code>mllib_VideoColorARGB2JFIFYCC444(3MLIB)</code> , <code>mllib_VideoColorRGB2JFIFYCC444(3MLIB)</code> , <code>attributes(5)</code>						

mllib_VideoColorRGBAint_to_ABGRint(3MLIB)

NAME mllib_VideoColorRGBAint_to_ABGRint – convert RGBA interleaved to ABGR

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_VideoColorRGBAint_to_ABGRint(mllib_u32 *ABGR, const
    mllib_u32 *RGBA, mllib_s32 w, mllib_s32 h, mllib_s32 dlb,
    mllib_s32 slb) ;
```

DESCRIPTION The RGBA pixel stream is broken apart and recombined into an ABGR pixel stream. All pixel components are 8-bit unsigned integers. The buffers have dimensions *w* and *h*. Within each 32-bit input word, the component ordering is R (bits 31-24), G (bits 23-16), B (bits 15-8), and A (bits 7-0). Within each 32-bit output word, the component ordering is A (bits 31-24), B (bits 23-16), G (bits 15-8), and R (bits 7-0).

PARAMETERS The function takes the following arguments:

ABGR Pointer to output buffer.

RGBA Pointer to input buffer.

w Image width in pixels.

h Image height in lines.

dlb Linebytes for output buffer.

slb Linebytes for input buffer.

RETURN VALUES None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_VideoColorABGRint_to_ARGBint(3MLIB)`,
`mllib_VideoColorBGRAint_to_ABGRint(3MLIB)`, `attributes(5)`

mllib_VideoColorRGBInt_to_ABGRInt(3MLIB)

NAME	mllib_VideoColorRGBInt_to_ABGRInt – convert RGB interleaved to ABGR interleaved																		
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> void mllib_VideoColorRGBInt_to_ABGRInt(mllib_u32 *ABGR, const mllib_u8 *RGB, const mllib_u8 *A_array, mllib_u8 A_const, mllib_s32 w, mllib_s32 h, mllib_s32 dlb, mllib_s32 slb, mllib_s32 alb);</pre>																		
DESCRIPTION	<p>The interleaved RGB stream, and the A values are combined into an A, B, G, R interleaved byte stream. Within each 24-bit input pixel, the component ordering is R (bits 23-16), G (bits 15-8), and B (bits 7-0). Within each 32-bit output word, the component ordering is A (bits 31-24), B (bits 23-16), G (bits 15-8), and R (bits 7-0).</p> <p>The alpha values for this function work in the following fashion:</p> <ul style="list-style-type: none">■ If <i>A_array</i> pointer is not NULL, the values are taken from there. It has to have the same dimensions as the R, G, and B buffers.■ If <i>A_array</i> pointer is NULL, the alpha values for every pixel are set to <i>A_const</i>.																		
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>ABGR</i></td><td>Pointer to output buffer.</td></tr><tr><td><i>RGB</i></td><td>Pointer to input buffer.</td></tr><tr><td><i>A_array</i></td><td>Array of alpha values.</td></tr><tr><td><i>A_const</i></td><td>Constant alpha value.</td></tr><tr><td><i>w</i></td><td>Image width in pixels.</td></tr><tr><td><i>h</i></td><td>Image height in lines.</td></tr><tr><td><i>dlb</i></td><td>Linebytes for output buffer.</td></tr><tr><td><i>slb</i></td><td>Linebytes for input buffer.</td></tr><tr><td><i>alb</i></td><td>Linebytes for alpha buffer.</td></tr></table>	<i>ABGR</i>	Pointer to output buffer.	<i>RGB</i>	Pointer to input buffer.	<i>A_array</i>	Array of alpha values.	<i>A_const</i>	Constant alpha value.	<i>w</i>	Image width in pixels.	<i>h</i>	Image height in lines.	<i>dlb</i>	Linebytes for output buffer.	<i>slb</i>	Linebytes for input buffer.	<i>alb</i>	Linebytes for alpha buffer.
<i>ABGR</i>	Pointer to output buffer.																		
<i>RGB</i>	Pointer to input buffer.																		
<i>A_array</i>	Array of alpha values.																		
<i>A_const</i>	Constant alpha value.																		
<i>w</i>	Image width in pixels.																		
<i>h</i>	Image height in lines.																		
<i>dlb</i>	Linebytes for output buffer.																		
<i>slb</i>	Linebytes for input buffer.																		
<i>alb</i>	Linebytes for alpha buffer.																		
RETURN VALUES	None.																		
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:																		

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

`mllib_VideoColorRGBint_to_ABGRint(3MLIB)`

SEE ALSO `mllib_VideoColorRGBseq_to_ABGRint(3MLIB),`
`mllib_VideoColorBGRint_to_ABGRint(3MLIB),`
`mllib_VideoColorRGBXint_to_ABGRint(3MLIB),`
`mllib_VideoColorRGBXint_to_ARGBint(3MLIB),`
`mllib_VideoColorXRGBint_to_ABGRint(3MLIB),`
`mllib_VideoColorXRGBint_to_ARGBint(3MLIB), attributes(5)`

mllib_VideoColorRGBseq_to_ABGRint(3MLIB)

NAME	mllib_VideoColorRGBseq_to_ABGRint – convert RGB sequential to ABGR interleaved																				
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> void mllib_VideoColorRGBseq_to_ABGRint(mllib_u32 *ABGR, const mllib_u8 *R, const mllib_u8 *G, const mllib_u8 *B, const mllib_u8 *A_array, mllib_u8 A_const, mllib_s32 w, mllib_s32 h, mllib_s32 dlb, mllib_s32 slb);</pre>																				
DESCRIPTION	<p>The R, G, and B streams, and the A values are combined into an A, B, G, R interleaved byte stream. Within each 32-bit output word, the component ordering is A (bits 31-24), B (bits 23-16), G (bits 15-8), and R (bits 7-0).</p> <p>The alpha values for this function work in the following fashion:</p> <ul style="list-style-type: none">■ If <i>A_array</i> pointer is not NULL, the values are taken from there. It has to have the same dimensions as the R, G, and B buffers.■ If <i>A_array</i> pointer is NULL, the alpha values for every pixel are set to <i>A_const</i>.																				
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>ABGR</i></td><td>Pointer to output buffer.</td></tr><tr><td><i>R</i></td><td>Pointer to input R buffer.</td></tr><tr><td><i>G</i></td><td>Pointer to input G buffer.</td></tr><tr><td><i>B</i></td><td>Pointer to input B buffer.</td></tr><tr><td><i>A_array</i></td><td>Array of alpha values.</td></tr><tr><td><i>A_const</i></td><td>Constant alpha value.</td></tr><tr><td><i>w</i></td><td>Image width in pixels.</td></tr><tr><td><i>h</i></td><td>Image height in lines.</td></tr><tr><td><i>dlb</i></td><td>Linebytes for output buffer.</td></tr><tr><td><i>slb</i></td><td>Linebytes for input buffers.</td></tr></table>	<i>ABGR</i>	Pointer to output buffer.	<i>R</i>	Pointer to input R buffer.	<i>G</i>	Pointer to input G buffer.	<i>B</i>	Pointer to input B buffer.	<i>A_array</i>	Array of alpha values.	<i>A_const</i>	Constant alpha value.	<i>w</i>	Image width in pixels.	<i>h</i>	Image height in lines.	<i>dlb</i>	Linebytes for output buffer.	<i>slb</i>	Linebytes for input buffers.
<i>ABGR</i>	Pointer to output buffer.																				
<i>R</i>	Pointer to input R buffer.																				
<i>G</i>	Pointer to input G buffer.																				
<i>B</i>	Pointer to input B buffer.																				
<i>A_array</i>	Array of alpha values.																				
<i>A_const</i>	Constant alpha value.																				
<i>w</i>	Image width in pixels.																				
<i>h</i>	Image height in lines.																				
<i>dlb</i>	Linebytes for output buffer.																				
<i>slb</i>	Linebytes for input buffers.																				
RETURN VALUES	None.																				
ATTRIBUTES	<p>See <code>attributes(5)</code> for descriptions of the following attributes:</p> <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe														
ATTRIBUTE TYPE	ATTRIBUTE VALUE																				
Interface Stability	Evolving																				
MT-Level	MT-Safe																				
SEE ALSO	mllib_VideoColorRGBint_to_ABGRint(3MLIB) , mllib_VideoColorBGRint_to_ABGRint(3MLIB) , mllib_VideoColorRGBXint_to_ABGRint(3MLIB) ,																				

`mllib_VideoColorRGBseq_to_ABGRint(3MLIB)`

```
mllib_VideoColorRGBxint_to_ARGBint(3MLIB),  
mllib_VideoColorXRGBint_to_ABGRint(3MLIB),  
mllib_VideoColorXRGBint_to_ARGBint(3MLIB), attributes(5)
```

mllib_VideoColorRGBXint_to_ABGRint(3MLIB)

NAME | mllib_VideoColorRGBXint_to_ABGRint – convert RGBX interleaved to ABGR interleaved

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_VideoColorRGBXint_to_ABGRint(mllib_u32 *ABGR, const
    mllib_u32 *RGBX, const mllib_u8 *A_array, mllib_u8 A_const,
    mllib_s32 w, mllib_s32 h, mllib_s32 dlb, mllib_s32 slb, mllib_s32
    alb);
```

DESCRIPTION | The interleaved RGBX stream and the alpha values are combined into an interleaved A, B, G, R output stream. Within each 32-bit input pixel, the component ordering is R (bits 31-24), G (bits 23-16), and B (bits 15-8). Within each 32-bit output pixel, the component bordering is A (bits 31-24), B (bits 23-16), G (bits 15-8), and R (bits 7-0). The alpha values for this function work in the following fashion: if *A_array* is not NULL, the values are taken from the corresponding locations in the alpha array, otherwise a constant alpha value, specified by *A_const*, is store in each output pixel. Each element in the alpha array is an unsigned byte. *w* and *h* define the dimensions of the region of the buffers to be processed. The linebyte parameters are used to advance the data pointers for each of the buffers.

PARAMETERS | The function takes the following arguments:

<i>ABGR</i>	Pointer to output buffer (word-aligned).
<i>RGBX</i>	Pointer to input buffer (word-aligned).
<i>A_array</i>	Pointer to array of alpha values (byte-aligned).
<i>A_const</i>	Constant alpha value (range = 0..255).
<i>w</i>	Image width in pixels.
<i>h</i>	Image height in lines.
<i>dlb</i>	Linebytes for output buffer.
<i>slb</i>	Linebytes for input buffer.
<i>alb</i>	Linebytes for alpha buffer.

RETURN VALUES | None.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

`mllib_VideoColorRGBXint_to_ABGRint(3MLIB)`

SEE ALSO `mllib_VideoColorRGBseq_to_ABGRint(3MLIB),`
`mllib_VideoColorRGBint_to_ABGRint(3MLIB),`
`mllib_VideoColorBGRint_to_ABGRint(3MLIB),`
`mllib_VideoColorRGBXint_to_ARGBint(3MLIB),`
`mllib_VideoColorXRGBint_to_ABGRint(3MLIB),`
`mllib_VideoColorXRGBint_to_ARGBint(3MLIB), attributes(5)`

mllib_VideoColorRGBXint_to_ARGBint(3MLIB)

NAME mllib_VideoColorRGBXint_to_ARGBint – convert RGBX interleaved to ARGB interleaved

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_VideoColorRGBXint_to_ARGBint(mllib_u32 *ARGB, const
    mllib_u32 *RGBX, const mllib_u8 *A_array, mllib_u8 A_const,
    mllib_s32 w, mllib_s32 h, mllib_s32 dlb, mllib_s32 slb, mllib_s32
    alb);
```

DESCRIPTION Similar to mllib_VideoColorRGBXint_to_ABGRint() except that the output component ordering is: A (bits 31-24), R (bits 23-16), G (bits 15-8), and B (bits 7-0).

PARAMETERS The function takes the following arguments:

ARGB Pointer to output buffer (word-aligned).

RGBX Pointer to input buffer (word-aligned).

A_array Pointer to array of alpha values (byte-aligned).

A_const Constant alpha value (range = 0..255).

w Image width in pixels.

h Image height in lines.

dlb Linebytes for output buffer.

slb Linebytes for input buffer.

alb Linebytes for alpha buffer.

RETURN VALUES None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorRGBseq_to_ABGRint(3MLIB), mllib_VideoColorRGBint_to_ABGRint(3MLIB), mllib_VideoColorBGRint_to_ABGRint(3MLIB), mllib_VideoColorRGBXint_to_ABGRint(3MLIB), mllib_VideoColorXRGBint_to_ABGRint(3MLIB), mllib_VideoColorXRGBint_to_ARGBint(3MLIB), attributes(5)

NAME mllib_VideoColorSplit2 – color conversion (color channel split)

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorSplit2(mllib_u8 *color1, mllib_u8 *color2,
    const mllib_u8 *colors, mllib_s32 n);
```

DESCRIPTION The mllib_VideoColorSplit2() function performs color channel split. The elements of the colors array are alternately copied into the color1 array and color2 array.

PARAMETERS The function takes the following arguments:

color1 Pointer to first color component row. color1 must be 8-byte aligned.

color2 Pointer to second color component row. color2 must be 8-byte aligned.

colors Pointer to colors multi-component row. colors must be 8-byte aligned.

n Length of color1 and color2 arrays. Length of colors must be 2*n.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorMerge2(3MLIB), mllib_VideoColorMerge2_S16(3MLIB), mllib_VideoColorMerge3(3MLIB), mllib_VideoColorMerge3_S16(3MLIB), mllib_VideoColorMerge4(3MLIB), mllib_VideoColorMerge4_S16(3MLIB), mllib_VideoColorSplit2_S16(3MLIB), mllib_VideoColorSplit3(3MLIB), mllib_VideoColorSplit3_S16(3MLIB), mllib_VideoColorSplit4(3MLIB), mllib_VideoColorSplit4_S16(3MLIB), attributes(5)

mllib_VideoColorSplit2_S16(3MLIB)

NAME mllib_VideoColorSplit2_S16 – color conversion (color channel split)

SYNOPSIS
cc [*flag...*] *file...* -lmllib [*library...*]
#include <mllib.h>

```
mllib_status mllib_VideoColorSplit2_S16(mllib_s16 *color1, mllib_s16
    *color2, const mllib_s16 *colors, mllib_s32 n);
```

DESCRIPTION The mllib_VideoColorSplit2_S16() function performs color channel split.

The elements of the colors array are alternately copied into the color1 array and color2 array.

PARAMETERS The function takes the following arguments:

<i>color1</i>	Pointer to first color component row. color1 must be 8-byte aligned.
<i>color2</i>	Pointer to second color component row. color2 must be 8-byte aligned.
<i>colors</i>	Pointer to colors multi-component row. colors must be 8-byte aligned.
<i>n</i>	Length of color1 and color2 arrays. Length of colors must be 2*n.

RETURN VALUES The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorMerge2(3MLIB), mllib_VideoColorMerge2_S16(3MLIB), mllib_VideoColorMerge3(3MLIB), mllib_VideoColorMerge3_S16(3MLIB), mllib_VideoColorMerge4(3MLIB), mllib_VideoColorMerge4_S16(3MLIB), mllib_VideoColorSplit2(3MLIB), mllib_VideoColorSplit3(3MLIB), mllib_VideoColorSplit3_S16(3MLIB), mllib_VideoColorSplit4(3MLIB), mllib_VideoColorSplit4_S16(3MLIB), attributes(5)

NAME mllib_VideoColorSplit3 – color conversion (color channel split)

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorSplit3(mllib_u8 *color1, mllib_u8 *color2,
    mllib_u8 *color3, const mllib_u8 *colors, mllib_s32 n);
```

DESCRIPTION The mllib_VideoColorSplit3() function performs color channel split.

The elements of the colors array are selected in consecutive groups of three. As each group is processed, the first element is stored in the color1 array; the second element, in the color2 array; and the third element, in the color3 array. This process is repeated until the end of the colors array is reached.

PARAMETERS The function takes the following arguments:

color1 Pointer to first color component row. color1 must be 8-byte aligned.

color2 Pointer to second color component row. color2 must be 8-byte aligned.

colors Pointer to colors multi-component row. colors must be 8-byte aligned.

n Length of color1, color2, and color3 arrays. Length of colors must be 3*n.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorMerge2(3MLIB), mllib_VideoColorMerge2_S16(3MLIB), mllib_VideoColorMerge3(3MLIB), mllib_VideoColorMerge3_S16(3MLIB), mllib_VideoColorMerge4(3MLIB), mllib_VideoColorMerge4_S16(3MLIB), mllib_VideoColorSplit2(3MLIB), mllib_VideoColorSplit2_S16(3MLIB), mllib_VideoColorSplit3_S16(3MLIB), mllib_VideoColorSplit4(3MLIB), mllib_VideoColorSplit4_S16(3MLIB), attributes(5)

mllib_VideoColorSplit3_S16(3MLIB)

NAME	mllib_VideoColorSplit3_S16 – color conversion (color channel split)								
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoColorSplit3_S16(mllib_s16 *color1, mllib_s16 *color2, mllib_s16 *color3, const mllib_s16 *colors, mllib_s32 n);</pre>								
DESCRIPTION	<p>The <code>mllib_VideoColorSplit3_S16()</code> function performs color channel split.</p> <p>The elements of the colors array are selected in consecutive groups of three. As each group is processed, the first element is stored in the color1 array; the second element, in the color2 array; and the third element, in the color3 array. This process is repeated until the end of the colors array is reached.</p>								
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>color1</i></td><td>Pointer to first color component row. <i>color1</i> must be 8-byte aligned.</td></tr><tr><td><i>color2</i></td><td>Pointer to second color component row. <i>color2</i> must be 8-byte aligned.</td></tr><tr><td><i>colors</i></td><td>Pointer to colors multi-component row. <i>colors</i> must be 8-byte aligned.</td></tr><tr><td><i>n</i></td><td>Length of <i>color1</i>, <i>color2</i>, and <i>color3</i> arrays. Length of <i>colors</i> must be 3*n.</td></tr></table>	<i>color1</i>	Pointer to first color component row. <i>color1</i> must be 8-byte aligned.	<i>color2</i>	Pointer to second color component row. <i>color2</i> must be 8-byte aligned.	<i>colors</i>	Pointer to colors multi-component row. <i>colors</i> must be 8-byte aligned.	<i>n</i>	Length of <i>color1</i> , <i>color2</i> , and <i>color3</i> arrays. Length of <i>colors</i> must be 3*n.
<i>color1</i>	Pointer to first color component row. <i>color1</i> must be 8-byte aligned.								
<i>color2</i>	Pointer to second color component row. <i>color2</i> must be 8-byte aligned.								
<i>colors</i>	Pointer to colors multi-component row. <i>colors</i> must be 8-byte aligned.								
<i>n</i>	Length of <i>color1</i> , <i>color2</i> , and <i>color3</i> arrays. Length of <i>colors</i> must be 3*n.								
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .								
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:								
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe		
ATTRIBUTE TYPE	ATTRIBUTE VALUE								
Interface Stability	Evolving								
MT-Level	MT-Safe								
SEE ALSO	<code>mllib_VideoColorMerge2(3MLIB)</code> , <code>mllib_VideoColorMerge2_S16(3MLIB)</code> , <code>mllib_VideoColorMerge3(3MLIB)</code> , <code>mllib_VideoColorMerge3_S16(3MLIB)</code> , <code>mllib_VideoColorMerge4(3MLIB)</code> , <code>mllib_VideoColorMerge4_S16(3MLIB)</code> , <code>mllib_VideoColorSplit2(3MLIB)</code> , <code>mllib_VideoColorSplit2_S16(3MLIB)</code> , <code>mllib_VideoColorSplit3(3MLIB)</code> , <code>mllib_VideoColorSplit4(3MLIB)</code> , <code>mllib_VideoColorSplit4_S16(3MLIB)</code> , <code>attributes(5)</code>								

NAME mllib_VideoColorSplit4 – color conversion (color channel split)

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorSplit4(mllib_u8 *color1, mllib_u8 *color2,
    mllib_u8 *color3, mllib_u8 *color4, const mllib_u8 *colors, mllib_s32
    n);
```

DESCRIPTION The mllib_VideoColorSplit4() function performs color channel split.

The elements of the colors array are selected in consecutive groups of four. As each group is processed, the first element is stored in the color1 array; the second element, in the color2 array; and so on. This process is repeated until the end of the colors array is reached.

PARAMETERS The function takes the following arguments:

color1 Pointer to first color component row. color1 must be 8-byte aligned.

color2 Pointer to second color component row. color2 must be 8-byte aligned.

color3 Pointer to third color component row. color3 must be 8-byte aligned.

color4 Pointer to fourth color component row. color4 must be 8-byte aligned.

colors Pointer to colors multi-component row. colors must be 8-byte aligned.

n Length of color1, color2, color3, and color4 arrays. Length of colors must be 4*n.

RETURN VALUES The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorMerge2(3MLIB), mllib_VideoColorMerge2_S16(3MLIB), mllib_VideoColorMerge3(3MLIB), mllib_VideoColorMerge3_S16(3MLIB), mllib_VideoColorMerge4(3MLIB), mllib_VideoColorMerge4_S16(3MLIB), mllib_VideoColorSplit2(3MLIB), mllib_VideoColorSplit2_S16(3MLIB), mllib_VideoColorSplit3(3MLIB), mllib_VideoColorSplit3_S16(3MLIB), mllib_VideoColorSplit4_S16(3MLIB), attributes(5)

mllib_VideoColorSplit4_S16(3MLIB)

NAME	mllib_VideoColorSplit4_S16 – color conversion (color channel split)								
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_VideoColorSplit4_S16(mllib_s16 *color1, mllib_s16 *color2, mllib_s16 *color3, mllib_s16 *color4, const mllib_s16 *colors, mllib_s32 n);</pre>								
DESCRIPTION	<p>The <code>mllib_VideoColorSplit4_S16()</code> function performs color channel split.</p> <p>The elements of the colors array are selected in consecutive groups of four. As each group is processed, the first element is stored in the color1 array; the second element, in the color2 array; and so on. This process is repeated until the end of the colors array is reached.</p>								
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>color1</i></td><td>Pointer to first color component row. color1 must be 8-byte aligned.</td></tr><tr><td><i>color2</i></td><td>Pointer to second color component row. color2 must be 8-byte aligned.</td></tr><tr><td><i>colors</i></td><td>Pointer to colors multi-component row. colors must be 8-byte aligned.</td></tr><tr><td><i>n</i></td><td>Length of color1, color2, color3, and color4 arrays. Length of colors must be 4*n.</td></tr></table>	<i>color1</i>	Pointer to first color component row. color1 must be 8-byte aligned.	<i>color2</i>	Pointer to second color component row. color2 must be 8-byte aligned.	<i>colors</i>	Pointer to colors multi-component row. colors must be 8-byte aligned.	<i>n</i>	Length of color1, color2, color3, and color4 arrays. Length of colors must be 4*n.
<i>color1</i>	Pointer to first color component row. color1 must be 8-byte aligned.								
<i>color2</i>	Pointer to second color component row. color2 must be 8-byte aligned.								
<i>colors</i>	Pointer to colors multi-component row. colors must be 8-byte aligned.								
<i>n</i>	Length of color1, color2, color3, and color4 arrays. Length of colors must be 4*n.								
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .								
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:								
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe		
ATTRIBUTE TYPE	ATTRIBUTE VALUE								
Interface Stability	Evolving								
MT-Level	MT-Safe								
SEE ALSO	<code>mllib_VideoColorMerge2(3MLIB)</code> , <code>mllib_VideoColorMerge2_S16(3MLIB)</code> , <code>mllib_VideoColorMerge3(3MLIB)</code> , <code>mllib_VideoColorMerge3_S16(3MLIB)</code> , <code>mllib_VideoColorMerge4(3MLIB)</code> , <code>mllib_VideoColorMerge4_S16(3MLIB)</code> , <code>mllib_VideoColorSplit2(3MLIB)</code> , <code>mllib_VideoColorSplit2_S16(3MLIB)</code> , <code>mllib_VideoColorSplit3(3MLIB)</code> , <code>mllib_VideoColorSplit3_S16(3MLIB)</code> , <code>mllib_VideoColorSplit4(3MLIB)</code> , <code>attributes(5)</code>								

mllib_VideoColorUYV444int_to_ABGRint(3MLIB)

NAME mllib_VideoColorUYV444int_to_ABGRint – color convert UYV interleaved to ABGR interleaved

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_VideoColorUYV444int_to_ABGRint(mllib_u32 *ABGR, const
    mllib_u8 *UYV, const mllib_u8 *A_array, mllib_u8 A_const, mllib_s32
    w, mllib_s32 h, mllib_s32 dlb, mllib_s32 slb, mllib_s32 alb);
```

DESCRIPTION

The UYV pixel stream is converted into an ABGR pixel stream. All pixel components are 8-bit unsigned integers. All buffers have dimensions *w* and *h*.

The alpha values for this function work in the following fashion:

- If *A_array* pointer is not NULL, the values are taken from there. It has to have the same dimensions as the Y buffer.
- If *A_array* pointer is NULL, the alpha values for every pixel are set to *A_const*.

The following equation is used:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.1644 & 0.0000 & 1.5966 \\ 1.1644 & -0.3920 & -0.8132 \\ 1.1644 & 2.0184 & 0.0000 \end{bmatrix} * \begin{bmatrix} Y \\ U \\ V \end{bmatrix} - \begin{bmatrix} 16.0000 \\ 128.0000 \\ 128.0000 \end{bmatrix}$$

PARAMETERS

The function takes the following arguments:

ABGR Pointer to output buffer.

UYV Pointer to input buffer.

A_array Array of alpha values.

A_const Constant alpha value.

w Image width in pixels.

h Image height in lines.

dlb Linebytes for output buffer.

slb Linebytes for input buffer.

alb Linebytes for alpha buffer.

RETURN VALUES None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

mllib_VideoColorUYV444int_to_ABGRint(3MLIB)

SEE ALSO | mllib_VideoColorYUV420seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV411seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV422seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV444seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV420seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV411seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV422seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV444seq_to_ABGRint(3MLIB),
mllib_VideoColorYUYV422int_to_ARGBint(3MLIB),
mllib_VideoColorYUV444int_to_ARGBint(3MLIB),
mllib_VideoColorYUYV422int_to_ABGRint(3MLIB),
mllib_VideoColorYUV444int_to_ABGRint(3MLIB),
mllib_VideoColorUYVY422int_to_ARGBint(3MLIB),
mllib_VideoColorUYVY422int_to_ABGRint(3MLIB),
mllib_VideoColorUYV444int_to_ARGBint(3MLIB), attributes(5)

mllib_VideoColorUYV444int_to_ARGBint(3MLIB)

NAME mllib_VideoColorUYV444int_to_ARGBint – color convert UYV interleaved to ARGB interleaved

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_VideoColorUYV444int_to_ARGBint(mllib_u32 *ARGB, const
      mllib_u8 *UYV, const mllib_u8 *A_array, mllib_u8 A_const, mllib_s32
      w, mllib_s32 h, mllib_s32 dlb, mllib_s32 slb, mllib_s32 alb);
```

DESCRIPTION The UYV pixel stream is converted into an ARGB pixel stream. All pixel components are 8-bit unsigned integers. All buffers have dimensions *w* and *h*.

The alpha values for this function work in the following fashion:

- If *A_array* pointer is not NULL, the values are taken from there. It has to have the same dimensions as the Y buffer.
- If *A_array* pointer is NULL, the alpha values for every pixel are set to *A_const*.

The following equation is used:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.1644 & 0.0000 & 1.5966 \\ 1.1644 & -0.3920 & -0.8132 \\ 1.1644 & 2.0184 & 0.0000 \end{bmatrix} * \begin{bmatrix} Y \\ U \\ V \end{bmatrix} - \begin{bmatrix} 16.0000 \\ 128.0000 \\ 128.0000 \end{bmatrix}$$

PARAMETERS The function takes the following arguments:

- ARGB* Pointer to output buffer.
- UYV* Pointer to input buffer.
- A_array* Array of alpha values.
- A_const* Constant alpha value.
- w* Image width in pixels.
- h* Image height in lines.
- dlb* Linebytes for output buffer.
- slb* Linebytes for input buffer.
- alb* Linebytes for alpha buffer.

RETURN VALUES None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

mllib_VideoColorUYV444int_to_ARGBint(3MLIB)

SEE ALSO | mllib_VideoColorYUV420seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV411seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV422seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV444seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV420seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV411seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV422seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV444seq_to_ABGRint(3MLIB),
mllib_VideoColorYUYV422int_to_ARGBint(3MLIB),
mllib_VideoColorYUV444int_to_ARGBint(3MLIB),
mllib_VideoColorYUYV422int_to_ABGRint(3MLIB),
mllib_VideoColorYUV444int_to_ABGRint(3MLIB),
mllib_VideoColorUYVY422int_to_ARGBint(3MLIB),
mllib_VideoColorUYVY422int_to_ABGRint(3MLIB),
mllib_VideoColorUYV444int_to_ABGRint(3MLIB), attributes(5)

mllib_VideoColorUYV444int_to_UYVY422int(3MLIB)

NAME mllib_VideoColorUYV444int_to_UYVY422int – convert UYV interleaved with subsampling

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_VideoColorUYV444int_to_UYVY422int(mllib_u32 *UYVY,
      const mllib_u8 *UYV, mllib_s32 w, mllib_s32 h, mllib_s32 dlb,
      mllib_s32 slb);
```

DESCRIPTION The UYV pixel stream is broken apart and recombined into a UYVY pixel stream. All pixel components are 8-bit unsigned integers. The UYV buffer has dimensions *w* and *h*. Dimension *w* is assumed to be a multiple of 2. Adjacent U and V values are averaged to get the output U and V values. The sequence of values in the input stream is U[r][c], Y[r][c], V[r][c], U[r][c+1], Y[r][c+1], V[r][c+1], ...

The following equation is used:

$$UYVY[r][c/2] = ((U[r][c] + U[r][c+1]) / 2) \ll 24 \mid (Y[r][c] \ll 16) \mid ((V[r][c] + V[r][c+1]) / 2) \ll 8 \mid (Y[r][c+1])$$

where *r* = 0, 1, 2, ..., *h*-1; and *c* = 0, 2, 4, ..., *w*-2.

PARAMETERS The function takes the following arguments:

UYVY Pointer to output buffer.

UYV Pointer to input buffer.

w Image width in pixels.

h Image height in lines.

dlb Linebytes for output buffer.

slb Linebytes for input buffer.

RETURN VALUES None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_VideoColorYUV444seq_to_YUYV422int\(3MLIB\)](#), [mllib_VideoColorYUV444int_to_YUYV422int\(3MLIB\)](#), [mllib_VideoColorYUV444seq_to_UYVY422int\(3MLIB\)](#), [mllib_VideoColorYUV444int_to_UYVY422int\(3MLIB\)](#), [mllib_VideoColorUYV444int_to_YUYV422int\(3MLIB\)](#), [attributes\(5\)](#)

mllib_VideoColorUYV444int_to_YUYV422int(3MLIB)

NAME | mllib_VideoColorUYV444int_to_YUYV422int – convert UYV interleaved with subsampling

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_VideoColorUYV444int_to_YUYV422int(mllib_u32 *YUYV,
      const mllib_u8 *UYV, mllib_s32 w, mllib_s32 h, mllib_s32 dlb,
      mllib_s32 slb);
```

DESCRIPTION | The UYV pixel stream is broken apart and recombined into a YUYV pixel stream. All pixel components are 8-bit unsigned integers. The UYV buffer has dimensions *w* and *h*. Dimension *w* is assumed to be a multiple of 2. Adjacent U and V values are averaged to get the output U and V values. The sequence of values in the input stream is U[r][c], Y[r][c], V[r][c], U[r][c+1], Y[r][c+1], V[r][c+1], ...

The following equation is used:

$$YUYV[r][c/2] = (Y[r][c] \ll 24) | ((U[r][c] + U[r][c+1]) / 2) \ll 16 | (Y[r][c+1] \ll 8) | ((V[r][c] + V[r][c+1]) / 2)$$

where *r* = 0, 1, 2, ..., *h*-1; and *c* = 0, 2, 4, ..., *w*-2.

PARAMETERS | The function takes the following arguments:

YUYV | Pointer to output buffer.

UYV | Pointer to input buffer.

w | Image width in pixels.

h | Image height in lines.

dlb | Linebytes for output buffer.

slb | Linebytes for input buffer.

RETURN VALUES | None.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_VideoColorYUV444seq_to_YUYV422int(3MLIB), mllib_VideoColorYUV444int_to_YUYV422int(3MLIB), mllib_VideoColorYUV444seq_to_UYVY422int(3MLIB), mllib_VideoColorYUV444int_to_UYVY422int(3MLIB), mllib_VideoColorUYV444int_to_UYVY422int(3MLIB), attributes(5)

mllib_VideoColorUYVY422int_to_ABGRint(3MLIB)

NAME mllib_VideoColorUYVY422int_to_ABGRint – color convert UYVY interleaved to ABGR interleaved

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_VideoColorUYVY422int_to_ABGRint(mllib_u32 *ABGR, const
      mllib_u32 *UYVY, const mllib_u8 *A_array, mllib_u8 A_const,
      mllib_s32 w, mllib_s32 h, mllib_s32 dlb, mllib_s32 slb, mllib_s32
      alb);
```

DESCRIPTION The UYVY pixel stream is converted into an ABGR pixel stream. All pixel components are 8-bit unsigned integers. The UYVY buffer has dimensions $w/2$ and h . The ABGR buffer has dimensions w and h . Dimension w is assumed to be even.

The alpha values for this function work in the following fashion:

- If A_array pointer is not NULL, the values are taken from there. It has to have the same dimensions as the Y buffer.
- If A_array pointer is NULL, the alpha values for every pixel are set to A_const .

The following equation is used:

$$\begin{bmatrix} |R| \\ |G| \\ |B| \end{bmatrix} = \begin{bmatrix} |1.1644 & 0.0000 & 1.5966| \\ |1.1644 & -0.3920 & -0.8132| \\ |1.1644 & 2.0184 & 0.0000| \end{bmatrix} * \begin{bmatrix} [|Y| \\ |U| \\ |V| \end{bmatrix} - \begin{bmatrix} |16.0000| \\ |128.0000| \\ |128.0000| \end{bmatrix}$$

PARAMETERS The function takes the following arguments:

ABGR Pointer to output buffer.

UYVY Pointer to input buffer.

A_array Array of alpha values.

A_const Constant alpha value.

w Image width in pixels.

h Image height in lines.

dlb Linebytes for output buffer.

slb Linebytes for input buffer.

alb Linebytes for alpha buffer.

RETURN VALUES None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

mllib_VideoColorUYVY422int_to_ABGRint(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	MT-Safe

SEE ALSO

mllib_VideoColorYUV420seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV411seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV422seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV444seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV420seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV411seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV422seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV444seq_to_ABGRint(3MLIB),
mllib_VideoColorYUYV422int_to_ARGBint(3MLIB),
mllib_VideoColorYUV444int_to_ARGBint(3MLIB),
mllib_VideoColorYUYV422int_to_ABGRint(3MLIB),
mllib_VideoColorYUV444int_to_ABGRint(3MLIB),
mllib_VideoColorUYVY422int_to_ARGBint(3MLIB),
mllib_VideoColorUYV444int_to_ARGBint(3MLIB),
mllib_VideoColorUYV444int_to_ABGRint(3MLIB), attributes(5)

mllib_VideoColorUYVY422int_to_ARGBint(3MLIB)

NAME mllib_VideoColorUYVY422int_to_ARGBint – color convert UYVY interleaved to ARGB interleaved

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_VideoColorUYVY422int_to_ARGBint(mllib_u32 *ARGB, const
    mllib_u32 *UYVY, const mllib_u8 *A_array, mllib_u8 A_const,
    mllib_s32 w, mllib_s32 h, mllib_s32 dlb, mllib_s32 slb, mllib_s32
    alb);
```

DESCRIPTION

The UYVY pixel stream is converted into an ARGB pixel stream. All pixel components are 8-bit unsigned integers. The UYVY buffer has dimensions $w/2$ and h . The ARGB buffer has dimensions w and h . Dimension w is assumed to be even.

The alpha values for this function work in the following fashion:

- If `A_array` pointer is not NULL, the values are taken from there. It has to have the same dimensions as the Y buffer.
- If `A_array` pointer is NULL, the alpha values for every pixel are set to `A_const`.

The following equation is used:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.1644 & 0.0000 & 1.5966 \\ 1.1644 & -0.3920 & -0.8132 \\ 1.1644 & 2.0184 & 0.0000 \end{bmatrix} * \begin{bmatrix} Y \\ U \\ V \end{bmatrix} - \begin{bmatrix} 16.0000 \\ 128.0000 \\ 128.0000 \end{bmatrix}$$

PARAMETERS

The function takes the following arguments:

`ARGB` Pointer to output buffer.

`UYVY` Pointer to input buffer.

`A_array` Array of alpha values.

`A_const` Constant alpha value.

`w` Image width in pixels.

`h` Image height in lines.

`dlb` Linebytes for output buffer.

`slb` Linebytes for input buffer.

`alb` Linebytes for alpha buffer.

RETURN VALUES None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

mllib_VideoColorUYVY422int_to_ARGBint(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	MT-Safe

SEE ALSO

mllib_VideoColorYUV420seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV411seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV422seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV444seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV420seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV411seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV422seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV444seq_to_ABGRint(3MLIB),
mllib_VideoColorYUYV422int_to_ARGBint(3MLIB),
mllib_VideoColorYUV444int_to_ARGBint(3MLIB),
mllib_VideoColorYUYV422int_to_ABGRint(3MLIB),
mllib_VideoColorYUV444int_to_ABGRint(3MLIB),
mllib_VideoColorUYVY422int_to_ABGRint(3MLIB),
mllib_VideoColorUYV444int_to_ARGBint(3MLIB),
mllib_VideoColorUYV444int_to_ABGRint(3MLIB), attributes(5)

NAME mllib_VideoColorXRGBint_to_ABGRint – convert XRGB interleaved to ABGR interleaved

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_VideoColorXRGBint_to_ABGRint(mllib_u32 *ABGR, const
    mllib_u32 *XRGB, const mllib_u8 *A_array, mllib_u8 A_const,
    mllib_s32 w, mllib_s32 h, mllib_s32 dlb, mllib_s32 slb, mllib_s32
    alb);
```

DESCRIPTION Similar to mllib_VideoColorRGBXint_to_ABGRint except that the input component ordering is: R (bits 23-16), G (bits 15-8), and B (bits 7-0).

PARAMETERS The function takes the following arguments:

ABGR Pointer to output buffer (word-aligned).

XRGB Pointer to input buffer (word-aligned).

A_array Pointer to array of alpha values (byte-aligned).

A_const Constant alpha value (range = 0..255).

w Image width in pixels.

h Image height in lines.

dlb Linebytes for output buffer.

slb Linebytes for input buffer.

alb Linebytes for alphabuffer.

RETURN VALUES None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorRGBseq_to_ABGRint(3MLIB),
 mllib_VideoColorRGBint_to_ABGRint(3MLIB),
 mllib_VideoColorBGRint_to_ABGRint(3MLIB),
 mllib_VideoColorRGBXint_to_ABGRint(3MLIB),
 mllib_VideoColorRGBXint_to_ARGBint(3MLIB),
 mllib_VideoColorXRGBint_to_ARGBint(3MLIB), attributes(5)

mllib_VideoColorXRGBint_to_ARGBint(3MLIB)

NAME	mllib_VideoColorXRGBint_to_ARGBint – convert XRGB interleaved to ARGB interleaved						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> void mllib_VideoColorXRGBint_to_ARGBint(mllib_u32 *ARGB, const mllib_u32 *XRGB, const mllib_u8 *A_array, mllib_u8 A_const, mllib_s32 w, mllib_s32 h, mllib_s32 dlb, mllib_s32 slb, mllib_s32 alb);</pre>						
DESCRIPTION	Similar to mllib_VideoColorRGBXint_to_ARGBint except that the input component ordering is: R (bits 23-16), G (bits 15-8), and B (bits 7-0).						
PARAMETERS	The function takes the following arguments: <i>ARGB</i> Pointer to output buffer (word-aligned). <i>XRGB</i> Pointer to input buffer (word-aligned). <i>A_array</i> Pointer to array of alpha values (byte-aligned). <i>A_const</i> Constant alpha value (range = 0..255). <i>w</i> Image width in pixels. <i>h</i> Image height in lines. <i>dlb</i> Linebytes for output buffer. <i>slb</i> Linebytes for input buffer. <i>alb</i> Linebytes for alphabuffer.						
RETURN VALUES	None.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_VideoColorRGBseq_to_ABGRint(3MLIB), mllib_VideoColorRGBint_to_ABGRint(3MLIB), mllib_VideoColorBGRint_to_ABGRint(3MLIB), mllib_VideoColorRGBXint_to_ABGRint(3MLIB), mllib_VideoColorRGBXint_to_ARGBint(3MLIB), mllib_VideoColorXRGBint_to_ABGRint(3MLIB), attributes(5)						

NAME mllib_VideoColorYUV2ABGR411 – YUV to RGB color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorYUV2ABGR411(mllib_u8 *abgr, const
mllib_u8 *y, const mllib_u8 *u, const mllib_u8 *v, mllib_s32 width,
mllib_s32 height, mllib_s32 rgb_stride, mllib_s32 y_stride, mllib_s32
uv_stride);
```

DESCRIPTION The mllib_VideoColorYUV2ABGR411 () function performs YUV to RGB color conversion used in digital video compression in the 4:1:1 sequence.

The luminance component is stored in Y, the chrominance components are stored in U and V, respectively. The size of the chrominance image depends on the chroma format used by the sequence. In this sequence, the chrominance components are subsampled 4-to-1 in only the horizontal direction in respect to the luminance component.

PARAMETERS The function takes the following arguments:

abgr Pointer to the destination packed ABGR image.

y Pointer to the source Y component.

u Pointer to the source U component.

v Pointer to the source V component.

width Width of the image.

height Height of the image.

rgb_stride Stride, in bytes, between adjacent rows in the destination image.

y_stride Stride, in bytes, between adjacent rows in the Y component image.

uv_stride Stride, in bytes, between adjacent rows in the U and V component images.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_VideoColorYUV2ABGR420\(3MLIB\)](#), [mllib_VideoColorYUV2ABGR422\(3MLIB\)](#), [mllib_VideoColorYUV2ABGR444\(3MLIB\)](#), [mllib_VideoColorYUV2ARGB411\(3MLIB\)](#),

mllib_VideoColorYUV2ABGR411(3MLIB)

```
mllib_VideoColorYUV2ARGB420(3MLIB),  
mllib_VideoColorYUV2ARGB422(3MLIB),  
mllib_VideoColorYUV2ARGB444(3MLIB),  
mllib_VideoColorYUV2RGB411(3MLIB), mllib_VideoColorYUV2RGB420(3MLIB),  
mllib_VideoColorYUV2RGB422(3MLIB), mllib_VideoColorYUV2RGB444(3MLIB),  
attributes(5)
```

NAME mllib_VideoColorYUV2ABGR420 – YUV to RGB color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorYUV2ABGR420(mllib_u8 *abgr, const
mllib_u8 *y, const mllib_u8 *u, const mllib_u8 *v, mllib_s32 width,
mllib_s32 height, mllib_s32 rgb_stride, mllib_s32 y_stride, mllib_s32
uv_stride);
```

DESCRIPTION The mllib_VideoColorYUV2ABGR420() function performs YUV to RGB color conversion used in MPEG1 and MPEG2 video compression in the 4:2:0 sequence.

The luminance component is stored in Y, the chrominance components are stored in U and V, respectively. The size of the chrominance image depends on the chroma format used by the sequence. In this sequence, the chrominance components are subsampled 2-to-1 in both the horizontal and vertical directions in respect to the luminance component.

PARAMETERS The function takes the following arguments:

abgr Pointer to the destination packed ABGR image.

y Pointer to the source Y component.

u Pointer to the source U component.

v Pointer to the source V component.

width Width of the image.

height Height of the image.

rgb_stride Stride, in bytes, between adjacent rows in the destination image.

y_stride Stride, in bytes, between adjacent rows in the Y component image.

uv_stride Stride, in bytes, between adjacent rows in the U and V component images.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

mllib_VideoColorYUV2ABGR420(3MLIB)

SEE ALSO | mllib_VideoColorYUV2ABGR411(3MLIB),
mllib_VideoColorYUV2ABGR422(3MLIB),
mllib_VideoColorYUV2ABGR444(3MLIB),
mllib_VideoColorYUV2ARGB411(3MLIB),
mllib_VideoColorYUV2ARGB420(3MLIB),
mllib_VideoColorYUV2ARGB422(3MLIB),
mllib_VideoColorYUV2ARGB444(3MLIB),
mllib_VideoColorYUV2RGB411(3MLIB), mllib_VideoColorYUV2RGB420(3MLIB),
mllib_VideoColorYUV2RGB422(3MLIB), mllib_VideoColorYUV2RGB444(3MLIB),
attributes(5)

NAME	mllib_VideoColorYUV2ABGR420_W – YUV to RGB color conversion																										
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_VideoColorYUV2ABGR420_W(mllib_u8 *<i>abgr</i>, const mllib_u8 *<i>y</i>, const mllib_u8 *<i>u</i>, const mllib_u8 *<i>v</i>, mllib_s32 <i>width</i>, mllib_s32 <i>height</i>, mllib_s32 <i>abgr_stride</i>, mllib_s32 <i>y_stride</i>, mllib_s32 <i>uv_stride</i>, mllib_s32 <i>left</i>, mllib_s32 <i>top</i>, mllib_s32 <i>right</i>, mllib_s32 <i>bottom</i>);</pre>																										
DESCRIPTION	<p>The <code>mllib_VideoColorYUV2ABGR420_W()</code> function performs YUV to RGB color conversion used in MPEG1 and MPEG2 video compression in the 4:2:0 sequence. It performs color conversion together with window clipping.</p> <p>The luminance component is stored in Y, the chrominance components are stored in U and V, respectively. The size of the chrominance image depends on the chroma format used by the sequence. In this sequence, the chrominance components are subsampled 2-to-1 in both the horizontal and vertical directions in respect to the luminance component.</p>																										
PARAMETERS	<p>The function takes the following arguments:</p> <table border="0"> <tr> <td style="padding-right: 20px;"><i>abgr</i></td> <td>Pointer to the destination packed ABGR image. <i>abgr</i> must be 8-byte aligned.</td> </tr> <tr> <td><i>y</i></td> <td>Pointer to the source Y component. <i>y</i> must be 8-byte aligned.</td> </tr> <tr> <td><i>u</i></td> <td>Pointer to the source U component. <i>u</i> must be 4-byte aligned.</td> </tr> <tr> <td><i>v</i></td> <td>Pointer to the source V component. <i>v</i> must be 4-byte aligned.</td> </tr> <tr> <td><i>width</i></td> <td>Width of the image. <i>width</i> must be a multiple of 8.</td> </tr> <tr> <td><i>height</i></td> <td>Height of the image. <i>height</i> must be a multiple of 2.</td> </tr> <tr> <td><i>abgr_stride</i></td> <td>Stride, in bytes, between adjacent rows in the ABGR image. <i>abgr_stride</i> must be a multiple of 8.</td> </tr> <tr> <td><i>y_stride</i></td> <td>Stride, in bytes, between adjacent rows in the Y component image. <i>y_stride</i> must be a multiple of 8.</td> </tr> <tr> <td><i>uv_stride</i></td> <td>Stride, in bytes, between adjacent rows in the U and V component images. <i>uv_stride</i> must be a multiple of 8.</td> </tr> <tr> <td><i>left</i></td> <td>Left border of clipping window. $0 \leq \textit{left} < \textit{right} \leq \textit{width}$.</td> </tr> <tr> <td><i>top</i></td> <td>Top border of clipping window. $0 \leq \textit{top} < \textit{bottom} \leq \textit{height}$.</td> </tr> <tr> <td><i>right</i></td> <td>Right border of clipping window. $0 \leq \textit{left} < \textit{right} \leq \textit{width}$.</td> </tr> <tr> <td><i>bottom</i></td> <td>Bottom border of clipping window. $0 \leq \textit{top} < \textit{bottom} \leq \textit{height}$.</td> </tr> </table>	<i>abgr</i>	Pointer to the destination packed ABGR image. <i>abgr</i> must be 8-byte aligned.	<i>y</i>	Pointer to the source Y component. <i>y</i> must be 8-byte aligned.	<i>u</i>	Pointer to the source U component. <i>u</i> must be 4-byte aligned.	<i>v</i>	Pointer to the source V component. <i>v</i> must be 4-byte aligned.	<i>width</i>	Width of the image. <i>width</i> must be a multiple of 8.	<i>height</i>	Height of the image. <i>height</i> must be a multiple of 2.	<i>abgr_stride</i>	Stride, in bytes, between adjacent rows in the ABGR image. <i>abgr_stride</i> must be a multiple of 8.	<i>y_stride</i>	Stride, in bytes, between adjacent rows in the Y component image. <i>y_stride</i> must be a multiple of 8.	<i>uv_stride</i>	Stride, in bytes, between adjacent rows in the U and V component images. <i>uv_stride</i> must be a multiple of 8.	<i>left</i>	Left border of clipping window. $0 \leq \textit{left} < \textit{right} \leq \textit{width}$.	<i>top</i>	Top border of clipping window. $0 \leq \textit{top} < \textit{bottom} \leq \textit{height}$.	<i>right</i>	Right border of clipping window. $0 \leq \textit{left} < \textit{right} \leq \textit{width}$.	<i>bottom</i>	Bottom border of clipping window. $0 \leq \textit{top} < \textit{bottom} \leq \textit{height}$.
<i>abgr</i>	Pointer to the destination packed ABGR image. <i>abgr</i> must be 8-byte aligned.																										
<i>y</i>	Pointer to the source Y component. <i>y</i> must be 8-byte aligned.																										
<i>u</i>	Pointer to the source U component. <i>u</i> must be 4-byte aligned.																										
<i>v</i>	Pointer to the source V component. <i>v</i> must be 4-byte aligned.																										
<i>width</i>	Width of the image. <i>width</i> must be a multiple of 8.																										
<i>height</i>	Height of the image. <i>height</i> must be a multiple of 2.																										
<i>abgr_stride</i>	Stride, in bytes, between adjacent rows in the ABGR image. <i>abgr_stride</i> must be a multiple of 8.																										
<i>y_stride</i>	Stride, in bytes, between adjacent rows in the Y component image. <i>y_stride</i> must be a multiple of 8.																										
<i>uv_stride</i>	Stride, in bytes, between adjacent rows in the U and V component images. <i>uv_stride</i> must be a multiple of 8.																										
<i>left</i>	Left border of clipping window. $0 \leq \textit{left} < \textit{right} \leq \textit{width}$.																										
<i>top</i>	Top border of clipping window. $0 \leq \textit{top} < \textit{bottom} \leq \textit{height}$.																										
<i>right</i>	Right border of clipping window. $0 \leq \textit{left} < \textit{right} \leq \textit{width}$.																										
<i>bottom</i>	Bottom border of clipping window. $0 \leq \textit{top} < \textit{bottom} \leq \textit{height}$.																										
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .																										

mllib_VideoColorYUV2ABGR420_W(3MLIB)

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_VideoColorYUV2ABGR420_WX2(3MLIB)`,
`mllib_VideoColorYUV2ABGR420_WX3(3MLIB)`,
`mllib_VideoColorYUV2ABGR420_X2(3MLIB)`,
`mllib_VideoColorYUV2ABGR420_X3(3MLIB)`, `attributes(5)`

mllib_VideoColorYUV2ABGR420_WX2(3MLIB)

NAME	mllib_VideoColorYUV2ABGR420_WX2 – YUV to RGB color conversion																										
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_VideoColorYUV2ABGR420_WX2(mllib_u8 *<i>abgr</i>, const mllib_u8 *<i>y</i>, const mllib_u8 *<i>u</i>, const mllib_u8 *<i>v</i>, mllib_s32 <i>width</i>, mllib_s32 <i>height</i>, mllib_s32 <i>abgr_stride</i>, mllib_s32 <i>y_stride</i>, mllib_s32 <i>uv_stride</i>, mllib_s32 <i>left</i>, mllib_s32 <i>top</i>, mllib_s32 <i>right</i>, mllib_s32 <i>bottom</i>);</pre>																										
DESCRIPTION	<p>The <code>mllib_VideoColorYUV2ABGR420_WX2()</code> function performs YUV to RGB color conversion used in MPEG1 and MPEG2 video compression in the 4:2:0 sequence. It performs color conversion together with window clipping and 2X zooming.</p> <p>The luminance component is stored in Y, the chrominance components are stored in U and V, respectively. The size of the chrominance image depends on the chroma format used by the sequence. In this sequence, the chrominance components are subsampled 2-to-1 in both the horizontal and vertical directions in respect to the luminance component.</p>																										
PARAMETERS	<p>The function takes the following arguments:</p> <table border="0" style="width: 100%;"> <tr> <td style="padding-right: 20px;"><i>abgr</i></td> <td>Pointer to the destination packed ABGR image. <i>abgr</i> must be 8-byte aligned.</td> </tr> <tr> <td><i>y</i></td> <td>Pointer to the source Y component. <i>y</i> must be 8-byte aligned.</td> </tr> <tr> <td><i>u</i></td> <td>Pointer to the source U component. <i>u</i> must be 4-byte aligned.</td> </tr> <tr> <td><i>v</i></td> <td>Pointer to the source V component. <i>v</i> must be 4-byte aligned.</td> </tr> <tr> <td><i>width</i></td> <td>Width of the image. <i>width</i> must be a multiple of 8.</td> </tr> <tr> <td><i>height</i></td> <td>Height of the image. <i>height</i> must be a multiple of 2.</td> </tr> <tr> <td><i>abgr_stride</i></td> <td>Stride, in bytes, between adjacent rows in the ABGR image. <i>abgr_stride</i> must be a multiple of 8.</td> </tr> <tr> <td><i>y_stride</i></td> <td>Stride, in bytes, between adjacent rows in the Y component image. <i>y_stride</i> must be a multiple of 8.</td> </tr> <tr> <td><i>uv_stride</i></td> <td>Stride, in bytes, between adjacent rows in the U and V component images. <i>uv_stride</i> must be a multiple of 8.</td> </tr> <tr> <td><i>left</i></td> <td>Left border of clipping window. $0 \leq \textit{left} < \textit{right} \leq \textit{width}$.</td> </tr> <tr> <td><i>top</i></td> <td>Top border of clipping window. $0 \leq \textit{top} < \textit{bottom} \leq \textit{height}$.</td> </tr> <tr> <td><i>right</i></td> <td>Left border of clipping window. $0 \leq \textit{left} < \textit{right} \leq \textit{width}$.</td> </tr> <tr> <td><i>bottom</i></td> <td>Bottom border of clipping window. $0 \leq \textit{top} < \textit{bottom} \leq \textit{height}$.</td> </tr> </table>	<i>abgr</i>	Pointer to the destination packed ABGR image. <i>abgr</i> must be 8-byte aligned.	<i>y</i>	Pointer to the source Y component. <i>y</i> must be 8-byte aligned.	<i>u</i>	Pointer to the source U component. <i>u</i> must be 4-byte aligned.	<i>v</i>	Pointer to the source V component. <i>v</i> must be 4-byte aligned.	<i>width</i>	Width of the image. <i>width</i> must be a multiple of 8.	<i>height</i>	Height of the image. <i>height</i> must be a multiple of 2.	<i>abgr_stride</i>	Stride, in bytes, between adjacent rows in the ABGR image. <i>abgr_stride</i> must be a multiple of 8.	<i>y_stride</i>	Stride, in bytes, between adjacent rows in the Y component image. <i>y_stride</i> must be a multiple of 8.	<i>uv_stride</i>	Stride, in bytes, between adjacent rows in the U and V component images. <i>uv_stride</i> must be a multiple of 8.	<i>left</i>	Left border of clipping window. $0 \leq \textit{left} < \textit{right} \leq \textit{width}$.	<i>top</i>	Top border of clipping window. $0 \leq \textit{top} < \textit{bottom} \leq \textit{height}$.	<i>right</i>	Left border of clipping window. $0 \leq \textit{left} < \textit{right} \leq \textit{width}$.	<i>bottom</i>	Bottom border of clipping window. $0 \leq \textit{top} < \textit{bottom} \leq \textit{height}$.
<i>abgr</i>	Pointer to the destination packed ABGR image. <i>abgr</i> must be 8-byte aligned.																										
<i>y</i>	Pointer to the source Y component. <i>y</i> must be 8-byte aligned.																										
<i>u</i>	Pointer to the source U component. <i>u</i> must be 4-byte aligned.																										
<i>v</i>	Pointer to the source V component. <i>v</i> must be 4-byte aligned.																										
<i>width</i>	Width of the image. <i>width</i> must be a multiple of 8.																										
<i>height</i>	Height of the image. <i>height</i> must be a multiple of 2.																										
<i>abgr_stride</i>	Stride, in bytes, between adjacent rows in the ABGR image. <i>abgr_stride</i> must be a multiple of 8.																										
<i>y_stride</i>	Stride, in bytes, between adjacent rows in the Y component image. <i>y_stride</i> must be a multiple of 8.																										
<i>uv_stride</i>	Stride, in bytes, between adjacent rows in the U and V component images. <i>uv_stride</i> must be a multiple of 8.																										
<i>left</i>	Left border of clipping window. $0 \leq \textit{left} < \textit{right} \leq \textit{width}$.																										
<i>top</i>	Top border of clipping window. $0 \leq \textit{top} < \textit{bottom} \leq \textit{height}$.																										
<i>right</i>	Left border of clipping window. $0 \leq \textit{left} < \textit{right} \leq \textit{width}$.																										
<i>bottom</i>	Bottom border of clipping window. $0 \leq \textit{top} < \textit{bottom} \leq \textit{height}$.																										
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .																										

mllib_VideoColorYUV2ABGR420_WX2(3MLIB)

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_VideoColorYUV2ABGR420_W(3MLIB)`,
`mllib_VideoColorYUV2ABGR420_WX3(3MLIB)`,
`mllib_VideoColorYUV2ABGR420_X2(3MLIB)`,
`mllib_VideoColorYUV2ABGR420_X3(3MLIB)`, `attributes(5)`

mllib_VideoColorYUV2ABGR420_WX3(3MLIB)

NAME	mllib_VideoColorYUV2ABGR420_WX3 – YUV to RGB color conversion																										
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoColorYUV2ABGR420_WX3(mllib_u8 *abgr, const mllib_u8 *y, const mllib_u8 *u, const mllib_u8 *v, mllib_s32 width, mllib_s32 height, mllib_s32 abgr_stride, mllib_s32 y_stride, mllib_s32 uv_stride, mllib_s32 left, mllib_s32 top, mllib_s32 right, mllib_s32 bottom) ;</pre>																										
DESCRIPTION	<p>The mllib_VideoColorYUV2ABGR420_WX3 () function performs YUV to RGB color conversion used in MPEG1 and MPEG2 video compression in the 4:2:0 sequence. It performs color conversion together with window clipping and 3X zooming.</p> <p>The luminance component is stored in Y, the chrominance components are stored in U and V, respectively. The size of the chrominance image depends on the chroma format used by the sequence. In this sequence, the chrominance components are subsampled 2-to-1 in both the horizontal and vertical directions in respect to the luminance component.</p>																										
PARAMETERS	<p>The function takes the following arguments:</p> <table border="0"> <tr> <td style="padding-right: 20px;"><i>abgr</i></td> <td>Pointer to the destination packed ABGR image. abgr must be 8-byte aligned.</td> </tr> <tr> <td><i>y</i></td> <td>Pointer to the source Y component. y must be 8-byte aligned.</td> </tr> <tr> <td><i>u</i></td> <td>Pointer to the source U component. u must be 4-byte aligned.</td> </tr> <tr> <td><i>v</i></td> <td>Pointer to the source V component. v must be 4-byte aligned.</td> </tr> <tr> <td><i>width</i></td> <td>Width of the image. width must be a multiple of 8.</td> </tr> <tr> <td><i>height</i></td> <td>Height of the image. height must be a multiple of 2.</td> </tr> <tr> <td><i>abgr_stride</i></td> <td>Stride, in bytes, between adjacent rows in the ABGR image. abgr_stride must be a multiple of 8.</td> </tr> <tr> <td><i>y_stride</i></td> <td>Stride, in bytes, between adjacent rows in the Y component image. y_stride must be a multiple of 8.</td> </tr> <tr> <td><i>uv_stride</i></td> <td>Stride, in bytes, between adjacent rows in the U and V component images. uv_stride must be a multiple of 8.</td> </tr> <tr> <td><i>left</i></td> <td>Left border of clipping window. $0 \leq \text{left} < \text{right} \leq \text{width}$.</td> </tr> <tr> <td><i>top</i></td> <td>Top border of clipping window. $0 \leq \text{top} < \text{bottom} \leq \text{height}$.</td> </tr> <tr> <td><i>right</i></td> <td>Right border of clipping window. $0 \leq \text{left} < \text{right} \leq \text{width}$.</td> </tr> <tr> <td><i>bottom</i></td> <td>Bottom border of clipping window. $0 \leq \text{top} < \text{bottom} \leq \text{height}$.</td> </tr> </table>	<i>abgr</i>	Pointer to the destination packed ABGR image. abgr must be 8-byte aligned.	<i>y</i>	Pointer to the source Y component. y must be 8-byte aligned.	<i>u</i>	Pointer to the source U component. u must be 4-byte aligned.	<i>v</i>	Pointer to the source V component. v must be 4-byte aligned.	<i>width</i>	Width of the image. width must be a multiple of 8.	<i>height</i>	Height of the image. height must be a multiple of 2.	<i>abgr_stride</i>	Stride, in bytes, between adjacent rows in the ABGR image. abgr_stride must be a multiple of 8.	<i>y_stride</i>	Stride, in bytes, between adjacent rows in the Y component image. y_stride must be a multiple of 8.	<i>uv_stride</i>	Stride, in bytes, between adjacent rows in the U and V component images. uv_stride must be a multiple of 8.	<i>left</i>	Left border of clipping window. $0 \leq \text{left} < \text{right} \leq \text{width}$.	<i>top</i>	Top border of clipping window. $0 \leq \text{top} < \text{bottom} \leq \text{height}$.	<i>right</i>	Right border of clipping window. $0 \leq \text{left} < \text{right} \leq \text{width}$.	<i>bottom</i>	Bottom border of clipping window. $0 \leq \text{top} < \text{bottom} \leq \text{height}$.
<i>abgr</i>	Pointer to the destination packed ABGR image. abgr must be 8-byte aligned.																										
<i>y</i>	Pointer to the source Y component. y must be 8-byte aligned.																										
<i>u</i>	Pointer to the source U component. u must be 4-byte aligned.																										
<i>v</i>	Pointer to the source V component. v must be 4-byte aligned.																										
<i>width</i>	Width of the image. width must be a multiple of 8.																										
<i>height</i>	Height of the image. height must be a multiple of 2.																										
<i>abgr_stride</i>	Stride, in bytes, between adjacent rows in the ABGR image. abgr_stride must be a multiple of 8.																										
<i>y_stride</i>	Stride, in bytes, between adjacent rows in the Y component image. y_stride must be a multiple of 8.																										
<i>uv_stride</i>	Stride, in bytes, between adjacent rows in the U and V component images. uv_stride must be a multiple of 8.																										
<i>left</i>	Left border of clipping window. $0 \leq \text{left} < \text{right} \leq \text{width}$.																										
<i>top</i>	Top border of clipping window. $0 \leq \text{top} < \text{bottom} \leq \text{height}$.																										
<i>right</i>	Right border of clipping window. $0 \leq \text{left} < \text{right} \leq \text{width}$.																										
<i>bottom</i>	Bottom border of clipping window. $0 \leq \text{top} < \text{bottom} \leq \text{height}$.																										
RETURN VALUES	The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.																										

mllib_VideoColorYUV2ABGR420_WX3(3MLIB)

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_VideoColorYUV2ABGR420_W(3MLIB)`,
`mllib_VideoColorYUV2ABGR420_WX2(3MLIB)`,
`mllib_VideoColorYUV2ABGR420_X2(3MLIB)`,
`mllib_VideoColorYUV2ABGR420_X3(3MLIB)`, `attributes(5)`

NAME mllib_VideoColorYUV2ABGR420_X2 – YUV to RGB color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorYUV2ABGR420_X2(mllib_u8 *abgr, const
mllib_u8 *y, const mllib_u8 *u, const mllib_u8 *v, mllib_s32 width,
mllib_s32 height, mllib_s32 abgr_stride, mllib_s32 y_stride, mllib_s32
uv_stride);
```

DESCRIPTION The mllib_VideoColorYUV2ABGR420_X2() function performs YUV to RGB color conversion used in MPEG1 and MPEG2 video compression in the 4:2:0 sequence. It performs color conversion together with 2X zooming.

The luminance component is stored in Y, the chrominance components are stored in U and V, respectively. The size of the chrominance image depends on the chroma format used by the sequence. In this sequence, the chrominance components are subsampled 2-to-1 in both the horizontal and vertical directions in respect to the luminance component.

PARAMETERS The function takes the following arguments:

abgr Pointer to the destination packed ABGR image. abgr must be 8-byte aligned.

y Pointer to the source Y component. y must be 8-byte aligned.

u Pointer to the source U component. u must be 4-byte aligned.

v Pointer to the source V component. v must be 4-byte aligned.

width Width of the image. width must be a multiple of 8.

height Height of the image. height must be a multiple of 2.

abgr_stride Stride, in bytes, between adjacent rows in the ABGR image. abgr_stride must be a multiple of 8.

y_stride Stride, in bytes, between adjacent rows in the Y component image. y_stride must be a multiple of 8.

uv_stride Stride, in bytes, between adjacent rows in the U and V component images. uv_stride must be a multiple of 8.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

mllib_VideoColorYUV2ABGR420_X2(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	MT-Safe

SEE ALSO [mllib_VideoColorYUV2ABGR420_W\(3MLIB\)](#),
[mllib_VideoColorYUV2ABGR420_WX2\(3MLIB\)](#),
[mllib_VideoColorYUV2ABGR420_WX3\(3MLIB\)](#),
[mllib_VideoColorYUV2ABGR420_X3\(3MLIB\)](#), [attributes\(5\)](#)

NAME mllib_VideoColorYUV2ABGR420_X3 – YUV to RGB color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorYUV2ABGR420_X3(mllib_u8 *abgr, const
mllib_u8 *y, const mllib_u8 *u, const mllib_u8 *v, mllib_s32 width,
mllib_s32 height, mllib_s32 abgr_stride, mllib_s32 y_stride, mllib_s32
uv_stride);
```

DESCRIPTION The `mllib_VideoColorYUV2ABGR420_X3()` function performs YUV to RGB color conversion used in MPEG1 and MPEG2 video compression in the 4:2:0 sequence. It performs color conversion together with 3X zooming.

The luminance component is stored in Y, the chrominance components are stored in U and V, respectively. The size of the chrominance image depends on the chroma format used by the sequence. In this sequence, the chrominance components are subsampled 2-to-1 in both the horizontal and vertical directions in respect to the luminance component.

PARAMETERS The function takes the following arguments:

abgr Pointer to the destination packed ABGR image. *abgr* must be 8-byte aligned.

y Pointer to the source Y component. *y* must be 8-byte aligned.

u Pointer to the source U component. *u* must be 4-byte aligned.

v Pointer to the source V component. *v* must be 4-byte aligned.

width Width of the image. *width* must be a multiple of 8.

height Height of the image. *height* must be a multiple of 2.

abgr_stride Stride, in bytes, between adjacent rows in the ABGR image. *abgr_stride* must be a multiple of 8.

y_stride Stride, in bytes, between adjacent rows in the Y component image. *y_stride* must be a multiple of 8.

uv_stride Stride, in bytes, between adjacent rows in the U and V component images. *uv_stride* must be a multiple of 8.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

mllib_VideoColorYUV2ABGR420_X3(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	MT-Safe

SEE ALSO [mllib_VideoColorYUV2ABGR420_W\(3MLIB\)](#),
[mllib_VideoColorYUV2ABGR420_WX2\(3MLIB\)](#),
[mllib_VideoColorYUV2ABGR420_WX3\(3MLIB\)](#),
[mllib_VideoColorYUV2ABGR420_X2\(3MLIB\)](#), [attributes\(5\)](#)

NAME mllib_VideoColorYUV2ABGR422 – YUV to RGB color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorYUV2ABGR422(mllib_u8 *abgr, const
mllib_u8 *y, const mllib_u8 *u, const mllib_u8 *v, mllib_s32 width,
mllib_s32 height, mllib_s32 rgb_stride, mllib_s32 y_stride, mllib_s32
uv_stride);
```

DESCRIPTION The mllib_VideoColorYUV2ABGR422() function performs YUV to RGB color conversion used in MPEG1 and MPEG2 video compression in the 4:2:2 sequence.

The luminance component is stored in Y, the chrominance components are stored in U and V, respectively. The size of the chrominance image depends on the chroma format used by the sequence. In this sequence, the chrominance components are subsampled 2-to-1 in only the horizontal direction in respect to the luminance component.

PARAMETERS The function takes the following arguments:

abgr Pointer to the destination packed ABGR image.

y Pointer to the source Y component.

u Pointer to the source U component.

v Pointer to the source V component.

width Width of the image.

height Height of the image.

rgb_stride Stride, in bytes, between adjacent rows in the destination image.

y_stride Stride, in bytes, between adjacent rows in the Y component image.

uv_stride Stride, in bytes, between adjacent rows in the U and V component images.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_VideoColorYUV2ABGR411\(3MLIB\)](#),
[mllib_VideoColorYUV2ABGR420\(3MLIB\)](#),
[mllib_VideoColorYUV2ABGR444\(3MLIB\)](#),
[mllib_VideoColorYUV2ARGB411\(3MLIB\)](#),

`mlib_VideoColorYUV2ABGR422(3MLIB)`

```
mllib_VideoColorYUV2ARGB420(3MLIB),  
mllib_VideoColorYUV2ARGB422(3MLIB),  
mllib_VideoColorYUV2ARGB444(3MLIB),  
mllib_VideoColorYUV2RGB411(3MLIB), mllib_VideoColorYUV2RGB420(3MLIB),  
mllib_VideoColorYUV2RGB422(3MLIB), mllib_VideoColorYUV2RGB444(3MLIB),  
attributes(5)
```

NAME mllib_VideoColorYUV2ABGR444 – YUV to RGB color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorYUV2ABGR444(mllib_u8 *abgr, const
mllib_u8 *y, const mllib_u8 *u, const mllib_u8 *v, mllib_s32 width,
mllib_s32 height, mllib_s32 rgb_stride, mllib_s32 yuv_stride);
```

DESCRIPTION The mllib_VideoColorYUV2ABGR444 () function performs YUV to RGB color conversion used in MPEG1 and MPEG2 video compression in the 4:4:4 sequence.

The luminance component is stored in Y, the chrominance components are stored in U and V, respectively. The size of the chrominance image depends on the chroma format used by the sequence. In this sequence, the chrominance components have the same resolution as the luminance component.

PARAMETERS The function takes the following arguments:

abgr Pointer to the destination packed ABGR image.

y Pointer to the source Y component.

u Pointer to the source U component.

v Pointer to the source V component.

width Width of the image.

height Height of the image.

rgb_stride Stride, in bytes, between adjacent rows in the destination image.

yuv_stride Stride, in bytes, between adjacent rows in the source image.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorYUV2ABGR411(3MLIB),
mllib_VideoColorYUV2ABGR420(3MLIB),
mllib_VideoColorYUV2ABGR422(3MLIB),
mllib_VideoColorYUV2ARGB411(3MLIB),
mllib_VideoColorYUV2ARGB420(3MLIB),

`mlib_VideoColorYUV2ABGR444(3MLIB)`

```
mllib_VideoColorYUV2ARGB422(3MLIB),  
mllib_VideoColorYUV2ARGB444(3MLIB),  
mllib_VideoColorYUV2RGB411(3MLIB), mllib_VideoColorYUV2RGB420(3MLIB),  
mllib_VideoColorYUV2RGB422(3MLIB), mllib_VideoColorYUV2RGB444(3MLIB),  
attributes(5)
```

NAME mllib_VideoColorYUV2ARGB411 – YUV to RGB color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorYUV2ARGB411(mllib_u8 *argb, const
mllib_u8 *y, const mllib_u8 *u, const mllib_u8 *v, mllib_s32 width,
mllib_s32 height, mllib_s32 rgb_stride, mllib_s32 y_stride, mllib_s32
uv_stride);
```

DESCRIPTION The `mllib_VideoColorYUV2ARGB411()` function performs YUV to RGB color conversion used in digital video compression in the 4:1:1 sequence.

The luminance component is stored in Y, the chrominance components are stored in U and V, respectively. The size of the chrominance image depends on the chroma format used by the sequence. In this sequence, the chrominance components are subsampled 4-to-1 in only the horizontal direction in respect to the luminance component.

PARAMETERS The function takes the following arguments:

argb Pointer to the destination packed ARGB image.

y Pointer to the source Y component.

u Pointer to the source U component.

v Pointer to the source V component.

width Width of the image.

height Height of the image.

rgb_stride Stride, in bytes, between adjacent rows in the destination image.

y_stride Stride, in bytes, between adjacent rows in the Y component image.

uv_stride Stride, in bytes, between adjacent rows in the U and V component images.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_VideoColorYUV2ABGR411\(3MLIB\)](#), [mllib_VideoColorYUV2ABGR420\(3MLIB\)](#), [mllib_VideoColorYUV2ABGR422\(3MLIB\)](#), [mllib_VideoColorYUV2ABGR444\(3MLIB\)](#),

mllib_VideoColorYUV2ARGB411(3MLIB)

```
mllib_VideoColorYUV2ARGB420(3MLIB),  
mllib_VideoColorYUV2ARGB422(3MLIB),  
mllib_VideoColorYUV2ARGB444(3MLIB),  
mllib_VideoColorYUV2RGB411(3MLIB), mllib_VideoColorYUV2RGB420(3MLIB),  
mllib_VideoColorYUV2RGB422(3MLIB), mllib_VideoColorYUV2RGB444(3MLIB),  
attributes(5)
```


mllib_VideoColorYUV2ARGB420(3MLIB)

NAME | mllib_VideoColorYUV2ARGB420 – YUV to RGB color conversion

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorYUV2ARGB420(mllib_u8 *argb, const
mllib_u8 *y, const mllib_u8 *u, const mllib_u8 *v, mllib_s32 width,
mllib_s32 height, mllib_s32 rgb_stride, mllib_s32 y_stride, mllib_s32
uv_stride);
```

DESCRIPTION | The `mllib_VideoColorYUV2ARGB420()` function performs YUV to RGB color conversion used in MPEG1 and MPEG2 video compression in the 4:2:0 sequence.

The luminance component is stored in Y, the chrominance components are stored in U and V, respectively. The size of the chrominance image depends on the chroma format used by the sequence. In this sequence, the chrominance components are subsampled 2-to-1 in both the horizontal and vertical directions in respect to the luminance component.

PARAMETERS | The function takes the following arguments:

argb | Pointer to the destination packed ARGB image.

y | Pointer to the source Y component.

u | Pointer to the source U component.

v | Pointer to the source V component.

width | Width of the image.

height | Height of the image.

rgb_stride | Stride, in bytes, between adjacent rows in the destination image.

y_stride | Stride, in bytes, between adjacent rows in the Y component image.

uv_stride | Stride, in bytes, between adjacent rows in the U and V component images.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

mllib_VideoColorYUV2ARGB420(3MLIB)

SEE ALSO | mllib_VideoColorYUV2ABGR411(3MLIB),
mllib_VideoColorYUV2ABGR420(3MLIB),
mllib_VideoColorYUV2ABGR422(3MLIB),
mllib_VideoColorYUV2ABGR444(3MLIB),
mllib_VideoColorYUV2ARGB411(3MLIB),
mllib_VideoColorYUV2ARGB422(3MLIB),
mllib_VideoColorYUV2ARGB444(3MLIB),
mllib_VideoColorYUV2RGB411(3MLIB), mllib_VideoColorYUV2RGB420(3MLIB),
mllib_VideoColorYUV2RGB422(3MLIB), mllib_VideoColorYUV2RGB444(3MLIB),
attributes(5)

NAME mllib_VideoColorYUV2ARGB422 – YUV to RGB color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorYUV2ARGB422(mllib_u8 *argb, const
mllib_u8 *y, const mllib_u8 *u, const mllib_u8 *v, mllib_s32 width,
mllib_s32 height, mllib_s32 rgb_stride, mllib_s32 y_stride, mllib_s32
uv_stride);
```

DESCRIPTION The mllib_VideoColorYUV2ARGB422() function performs YUV to RGB color conversion used in MPEG1 and MPEG2 video compression in the 4:2:2 sequence.

The luminance component is stored in Y, the chrominance components are stored in U and V, respectively. The size of the chrominance image depends on the chroma format used by the sequence. In this sequence, the chrominance components are subsampled 2-to-1 in only the horizontal direction in respect to the luminance component.

PARAMETERS The function takes the following arguments:

argb Pointer to the destination packed ARGB image.

y Pointer to the source Y component.

u Pointer to the source U component.

v Pointer to the source V component.

width Width of the image.

height Height of the image.

rgb_stride Stride, in bytes, between adjacent rows in the destination image.

y_stride Stride, in bytes, between adjacent rows in the Y component image.

uv_stride Stride, in bytes, between adjacent rows in the U and V component images.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_VideoColorYUV2ABGR411\(3MLIB\)](#),
[mllib_VideoColorYUV2ABGR420\(3MLIB\)](#),
[mllib_VideoColorYUV2ABGR422\(3MLIB\)](#),
[mllib_VideoColorYUV2ABGR444\(3MLIB\)](#),

mllib_VideoColorYUV2ARGB422(3MLIB)

```
mllib_VideoColorYUV2ARGB411(3MLIB),  
mllib_VideoColorYUV2ARGB420(3MLIB),  
mllib_VideoColorYUV2ARGB444(3MLIB),  
mllib_VideoColorYUV2RGB411(3MLIB), mllib_VideoColorYUV2RGB420(3MLIB),  
mllib_VideoColorYUV2RGB422(3MLIB), mllib_VideoColorYUV2RGB444(3MLIB),  
attributes(5)
```

mllib_VideoColorYUV2ARGB444(3MLIB)

NAME mllib_VideoColorYUV2ARGB444 – YUV to RGB color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorYUV2ARGB444(mllib_u8 *argb, const
mllib_u8 *y, const mllib_u8 *u, const mllib_u8 *v, mllib_s32 width,
mllib_s32 height, mllib_s32 rgb_stride, mllib_s32 yuv_stride);
```

DESCRIPTION The mllib_VideoColorYUV2ARGB444 () function performs YUV to RGB color conversion used in MPEG1 and MPEG2 video compression in the 4:4:4 sequence.

The luminance component is stored in Y, the chrominance components are stored in U and V, respectively. The size of the chrominance image depends on the chroma format used by the sequence. In this sequence, the chrominance components have the same resolution as the luminance component.

PARAMETERS The function takes the following arguments:

argb Pointer to the destination packed ARGB image.

y Pointer to the source Y component.

u Pointer to the source U component.

v Pointer to the source V component.

width Width of the image.

height Height of the image.

rgb_stride Stride, in bytes, between adjacent rows in the destination image.

yuv_stride Stride, in bytes, between adjacent rows in the source image.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorYUV2ABGR411(3MLIB),
mllib_VideoColorYUV2ABGR420(3MLIB),
mllib_VideoColorYUV2ABGR422(3MLIB),
mllib_VideoColorYUV2ABGR444(3MLIB),
mllib_VideoColorYUV2ARGB411(3MLIB),

`mlib_VideoColorYUV2ARGB444(3MLIB)`

```
mllib_VideoColorYUV2ARGB420(3MLIB),  
mllib_VideoColorYUV2ARGB422(3MLIB),  
mllib_VideoColorYUV2RGB411(3MLIB), mllib_VideoColorYUV2RGB420(3MLIB),  
mllib_VideoColorYUV2RGB422(3MLIB), mllib_VideoColorYUV2RGB444(3MLIB),  
attributes(5)
```

NAME mllib_VideoColorYUV2RGB411 – YUV to RGB color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorYUV2RGB411(mllib_u8 *rgb, const mllib_u8
    *y, const mllib_u8 *u, const mllib_u8 *v, mllib_s32 width,
    mllib_s32 height, mllib_s32 rgb_stride, mllib_s32 y_stride, mllib_s32
    uv_stride);
```

DESCRIPTION The mllib_VideoColorYUV2RGB411() function performs YUV to RGB color conversion used in digital video compression in the 4:1:1 sequence.

The luminance component is stored in Y, the chrominance components are stored in U and V, respectively. The size of the chrominance image depends on the chroma format used by the sequence. In this sequence, the chrominance components are subsampled 4-to-1 in only the horizontal direction in respect to the luminance component.

PARAMETERS The function takes the following arguments:

rgb Pointer to the destination RGB image.

y Pointer to the source Y component.

u Pointer to the source U component.

v Pointer to the source V component.

width Width of the image.

height Height of the image.

rgb_stride Stride, in bytes, between adjacent rows in the destination image.

y_stride Stride, in bytes, between adjacent rows in the Y component image.

uv_stride Stride, in bytes, between adjacent rows in the U and V component images.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_VideoColorYUV2ABGR411\(3MLIB\)](#),
[mllib_VideoColorYUV2ABGR420\(3MLIB\)](#),
[mllib_VideoColorYUV2ABGR422\(3MLIB\)](#),
[mllib_VideoColorYUV2ABGR444\(3MLIB\)](#),

mllib_VideoColorYUV2RGB411(3MLIB)

```
mllib_VideoColorYUV2ARGB411(3MLIB),  
mllib_VideoColorYUV2ARGB420(3MLIB),  
mllib_VideoColorYUV2ARGB422(3MLIB),  
mllib_VideoColorYUV2ARGB444(3MLIB),  
mllib_VideoColorYUV2RGB420(3MLIB), mllib_VideoColorYUV2RGB422(3MLIB),  
mllib_VideoColorYUV2RGB444(3MLIB), attributes(5)
```


NAME mllib_VideoColorYUV2RGB420 – YUV to RGB color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorYUV2RGB420(mllib_u8 *rgb, const mllib_u8
    *y, const mllib_u8 *u, const mllib_u8 *v, mllib_s32 width,
    mllib_s32 height, mllib_s32 rgb_stride, mllib_s32 y_stride, mllib_s32
    uv_stride);
```

DESCRIPTION The mllib_VideoColorYUV2RGB420() function performs YUV to RGB color conversion used in MPEG1 and MPEG2 video compression in the 4:2:0 sequence.

The luminance component is stored in Y, the chrominance components are stored in U and V, respectively. The size of the chrominance image depends on the chroma format used by the sequence. In this sequence, the chrominance components are subsampled 2-to-1 in both the horizontal and vertical directions in respect to the luminance component.

PARAMETERS The function takes the following arguments:

rgb Pointer to the destination RGB image.

y Pointer to the source Y component.

u Pointer to the source U component.

v Pointer to the source V component.

width Width of the image.

height Height of the image.

rgb_stride Stride, in bytes, between adjacent rows in the destination image.

y_stride Stride, in bytes, between adjacent rows in the Y component image.

uv_stride Stride, in bytes, between adjacent rows in the U and V component images.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

mllib_VideoColorYUV2RGB420(3MLIB)

SEE ALSO | mllib_VideoColorYUV2ABGR411(3MLIB),
mllib_VideoColorYUV2ABGR420(3MLIB),
mllib_VideoColorYUV2ABGR422(3MLIB),
mllib_VideoColorYUV2ABGR444(3MLIB),
mllib_VideoColorYUV2ARGB411(3MLIB),
mllib_VideoColorYUV2ARGB420(3MLIB),
mllib_VideoColorYUV2ARGB422(3MLIB),
mllib_VideoColorYUV2ARGB444(3MLIB),
mllib_VideoColorYUV2RGB411(3MLIB), mllib_VideoColorYUV2RGB422(3MLIB),
mllib_VideoColorYUV2RGB444(3MLIB), attributes(5)

NAME mllib_VideoColorYUV2RGB422 – YUV to RGB color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorYUV2RGB422(mllib_u8 *rgb, const mllib_u8
    *y, const mllib_u8 *u, const mllib_u8 *v, mllib_s32 width,
    mllib_s32 height, mllib_s32 rgb_stride, mllib_s32 y_stride, mllib_s32
    uv_stride);
```

DESCRIPTION The mllib_VideoColorYUV2RGB422() function performs YUV to RGB color conversion used in MPEG1 and MPEG2 video compression in the 4:2:2 sequence.

The luminance component is stored in Y, the chrominance components are stored in U and V, respectively. The size of the chrominance image depends on the chroma format used by the sequence. In this sequence, the chrominance components are subsampled 2-to-1 in only the horizontal direction in respect to the luminance component.

PARAMETERS The function takes the following arguments:

rgb Pointer to the destination RGB image.

y Pointer to the source Y component.

u Pointer to the source U component.

v Pointer to the source V component.

width Width of the image.

height Height of the image.

rgb_stride Stride, in bytes, between adjacent rows in the destination image.

y_stride Stride, in bytes, between adjacent rows in the Y component image.

uv_stride Stride, in bytes, between adjacent rows in the U and V component images.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_VideoColorYUV2ABGR411\(3MLIB\)](#),
[mllib_VideoColorYUV2ABGR420\(3MLIB\)](#),
[mllib_VideoColorYUV2ABGR422\(3MLIB\)](#),
[mllib_VideoColorYUV2ABGR444\(3MLIB\)](#),

`mllib_VideoColorYUV2RGB422(3MLIB)`

```
mllib_VideoColorYUV2ARGB411(3MLIB),  
mllib_VideoColorYUV2ARGB420(3MLIB),  
mllib_VideoColorYUV2ARGB422(3MLIB),  
mllib_VideoColorYUV2ARGB444(3MLIB),  
mllib_VideoColorYUV2RGB411(3MLIB), mllib_VideoColorYUV2RGB420(3MLIB),  
mllib_VideoColorYUV2RGB444(3MLIB), attributes(5)
```

NAME mllib_VideoColorYUV2RGB444 – YUV to RGB color conversion

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoColorYUV2RGB444(mllib_u8 *rgb, const mllib_u8
    *y, const mllib_u8 *u, const mllib_u8 *v, mllib_s32 width,
    mllib_s32 height, mllib_s32 rgb_stride, mllib_s32 yuv_stride);
```

DESCRIPTION The mllib_VideoColorYUV2RGB444 () function performs YUV to RGB color conversion used in MPEG1 and MPEG2 video compression in the 4:4:4 sequence.

The luminance component is stored in Y, the chrominance components are stored in U and V, respectively. The size of the chrominance image depends on the chroma format used by the sequence. In this sequence, the chrominance components have the same resolution as the luminance component.

PARAMETERS The function takes the following arguments:

rgb Pointer to the destination RGB image.

y Pointer to the source Y component.

u Pointer to the source U component.

v Pointer to the source V component.

width Width of the image.

height Height of the image.

rgb_stride Stride, in bytes, between adjacent rows in the destination image.

yuv_stride Stride, in bytes, between adjacent rows in the source image.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorYUV2ABGR411(3MLIB),
 mllib_VideoColorYUV2ABGR420(3MLIB),
 mllib_VideoColorYUV2ABGR422(3MLIB),
 mllib_VideoColorYUV2ABGR444(3MLIB),
 mllib_VideoColorYUV2ARGB411(3MLIB),
 mllib_VideoColorYUV2ARGB420(3MLIB),

`mlib_VideoColorYUV2RGB444(3MLIB)`

```
mlib_VideoColorYUV2ARGB422(3MLIB),  
mlib_VideoColorYUV2ARGB444(3MLIB),  
mlib_VideoColorYUV2RGB411(3MLIB), mlib_VideoColorYUV2RGB420(3MLIB),  
mlib_VideoColorYUV2RGB422(3MLIB), attributes(5)
```

mllib_VideoColorYUV411seq_to_ABGRint(3MLIB)

NAME	mllib_VideoColorYUV411seq_to_ABGRint – color convert YUV sequential to ABGR interleaved																						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> void mllib_VideoColorYUV411seq_to_ABGRint(mllib_u32 *ABGR, const mllib_u8 *Y, const mllib_u8 *U, const mllib_u8 *V, const mllib_u8 *A_array, mllib_u8 A_const, mllib_s32 w, mllib_s32 h, mllib_s32 dlb, mllib_s32 yalb, mllib_s32 uvlb);</pre>																						
DESCRIPTION	<p>The Y, U, V pixel streams are converted into an ABGR pixel stream. All pixel components are 8-bit unsigned integers. The Y buffer has dimensions <i>w</i> and <i>h</i>. The U and V buffers have dimensions <i>w</i>/4 and <i>h</i>. Dimension <i>w</i> is assumed to be a multiple of 4. In each row, every 4 Y values use the same U and V values.</p> <p>The alpha values for this function work in the following fashion:</p> <ul style="list-style-type: none"> ■ If <i>A_array</i> pointer is not NULL, the values are taken from there. It has to have the same dimensions as the Y buffer. ■ If <i>A_array</i> pointer is NULL, the alpha values for every pixel are set to <i>A_const</i>. <p>The following equation is used:</p> $\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.1644 & 0.0000 & 1.5966 \\ 1.1644 & -0.3920 & -0.8132 \\ 1.1644 & 2.0184 & 0.0000 \end{bmatrix} * \begin{bmatrix} Y \\ U \\ V \end{bmatrix} - \begin{bmatrix} 16.0000 \\ 128.0000 \\ 128.0000 \end{bmatrix}$																						
PARAMETERS	<p>The function takes the following arguments:</p> <table border="0" style="width: 100%;"> <tr> <td style="padding-right: 20px;"><i>ABGR</i></td> <td>Pointer to output buffer.</td> </tr> <tr> <td><i>Y</i></td> <td>Pointer to Y input buffer.</td> </tr> <tr> <td><i>U</i></td> <td>Pointer to U input buffer.</td> </tr> <tr> <td><i>V</i></td> <td>Pointer to V input buffer.</td> </tr> <tr> <td><i>A_array</i></td> <td>Array of alpha values.</td> </tr> <tr> <td><i>A_const</i></td> <td>Constant alpha value.</td> </tr> <tr> <td><i>w</i></td> <td>Image width in pixels.</td> </tr> <tr> <td><i>h</i></td> <td>Image height in lines.</td> </tr> <tr> <td><i>dlb</i></td> <td>Linebytes for output buffer.</td> </tr> <tr> <td><i>yalb</i></td> <td>Linebytes for Y and alpha buffers.</td> </tr> <tr> <td><i>uvlb</i></td> <td>Linebytes for U and V buffers.</td> </tr> </table>	<i>ABGR</i>	Pointer to output buffer.	<i>Y</i>	Pointer to Y input buffer.	<i>U</i>	Pointer to U input buffer.	<i>V</i>	Pointer to V input buffer.	<i>A_array</i>	Array of alpha values.	<i>A_const</i>	Constant alpha value.	<i>w</i>	Image width in pixels.	<i>h</i>	Image height in lines.	<i>dlb</i>	Linebytes for output buffer.	<i>yalb</i>	Linebytes for Y and alpha buffers.	<i>uvlb</i>	Linebytes for U and V buffers.
<i>ABGR</i>	Pointer to output buffer.																						
<i>Y</i>	Pointer to Y input buffer.																						
<i>U</i>	Pointer to U input buffer.																						
<i>V</i>	Pointer to V input buffer.																						
<i>A_array</i>	Array of alpha values.																						
<i>A_const</i>	Constant alpha value.																						
<i>w</i>	Image width in pixels.																						
<i>h</i>	Image height in lines.																						
<i>dlb</i>	Linebytes for output buffer.																						
<i>yalb</i>	Linebytes for Y and alpha buffers.																						
<i>uvlb</i>	Linebytes for U and V buffers.																						
RETURN VALUES	None.																						

mllib_VideoColorYUV411seq_to_ABGRint(3MLIB)

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorYUV420seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV411seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV422seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV444seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV420seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV422seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV444seq_to_ABGRint(3MLIB),
mllib_VideoColorYUYV422int_to_ARGBint(3MLIB),
mllib_VideoColorYUV444int_to_ARGBint(3MLIB),
mllib_VideoColorYUYV422int_to_ABGRint(3MLIB),
mllib_VideoColorYUV444int_to_ABGRint(3MLIB),
mllib_VideoColorUYVY422int_to_ARGBint(3MLIB),
mllib_VideoColorUYVY422int_to_ABGRint(3MLIB),
mllib_VideoColorUYV444int_to_ARGBint(3MLIB),
mllib_VideoColorUYV444int_to_ABGRint(3MLIB), attributes(5)

mllib_VideoColorYUV411seq_to_ARGBint(3MLIB)

NAME	mllib_VideoColorYUV411seq_to_ARGBint – color convert YUV sequential to ARGB interleaved																						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> void mllib_VideoColorYUV411seq_to_ARGBint(mllib_u32 *ARGB, const mllib_u8 *Y, const mllib_u8 *U, const mllib_u8 *V, const mllib_u8 *A_array, mllib_u8 A_const, mllib_s32 w, mllib_s32 h, mllib_s32 dlb, mllib_s32 yalb, mllib_s32 uvlb);</pre>																						
DESCRIPTION	<p>The Y, U, V pixel streams are converted into an ARGB pixel stream. All pixel components are 8-bit unsigned integers. The Y buffer has dimensions <i>w</i> and <i>h</i>. The U and V buffers have dimensions <i>w</i>/4 and <i>h</i>. Dimension <i>w</i> is assumed to be a multiple of 4. In each row, every 4 Y values use the same U and V values.</p> <p>The alpha values for this function work in the following fashion:</p> <ul style="list-style-type: none">■ If <i>A_array</i> pointer is not NULL, the values are taken from there. It has to have the same dimensions as the Y buffer.■ If <i>A_array</i> pointer is NULL, the alpha values for every pixel are set to <i>A_const</i>. <p>The following equation is used:</p> $\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.1644 & 0.0000 & 1.5966 \\ 1.1644 & -0.3920 & -0.8132 \\ 1.1644 & 2.0184 & 0.0000 \end{bmatrix} * \begin{bmatrix} Y \\ U \\ V \end{bmatrix} - \begin{bmatrix} 16.0000 \\ 128.0000 \\ 128.0000 \end{bmatrix}$																						
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>ARGB</i></td><td>Pointer to output buffer.</td></tr><tr><td><i>Y</i></td><td>Pointer to Y input buffer.</td></tr><tr><td><i>U</i></td><td>Pointer to U input buffer.</td></tr><tr><td><i>V</i></td><td>Pointer to V input buffer.</td></tr><tr><td><i>A_array</i></td><td>Array of alpha values.</td></tr><tr><td><i>A_const</i></td><td>Constant alpha value.</td></tr><tr><td><i>w</i></td><td>Image width in pixels.</td></tr><tr><td><i>h</i></td><td>Image height in lines.</td></tr><tr><td><i>dlb</i></td><td>Linebytes for output buffer.</td></tr><tr><td><i>yalb</i></td><td>Linebytes for Y and alpha buffers.</td></tr><tr><td><i>uvlb</i></td><td>Linebytes for U and V buffers.</td></tr></table>	<i>ARGB</i>	Pointer to output buffer.	<i>Y</i>	Pointer to Y input buffer.	<i>U</i>	Pointer to U input buffer.	<i>V</i>	Pointer to V input buffer.	<i>A_array</i>	Array of alpha values.	<i>A_const</i>	Constant alpha value.	<i>w</i>	Image width in pixels.	<i>h</i>	Image height in lines.	<i>dlb</i>	Linebytes for output buffer.	<i>yalb</i>	Linebytes for Y and alpha buffers.	<i>uvlb</i>	Linebytes for U and V buffers.
<i>ARGB</i>	Pointer to output buffer.																						
<i>Y</i>	Pointer to Y input buffer.																						
<i>U</i>	Pointer to U input buffer.																						
<i>V</i>	Pointer to V input buffer.																						
<i>A_array</i>	Array of alpha values.																						
<i>A_const</i>	Constant alpha value.																						
<i>w</i>	Image width in pixels.																						
<i>h</i>	Image height in lines.																						
<i>dlb</i>	Linebytes for output buffer.																						
<i>yalb</i>	Linebytes for Y and alpha buffers.																						
<i>uvlb</i>	Linebytes for U and V buffers.																						
RETURN VALUES	None.																						

mllib_VideoColorYUV411seq_to_ARGBint(3MLIB)

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorYUV420seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV422seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV444seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV420seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV411seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV422seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV444seq_to_ABGRint(3MLIB),
mllib_VideoColorYUYV422int_to_ARGBint(3MLIB),
mllib_VideoColorYUV444int_to_ARGBint(3MLIB),
mllib_VideoColorYUYV422int_to_ABGRint(3MLIB),
mllib_VideoColorYUV444int_to_ABGRint(3MLIB),
mllib_VideoColorUYVY422int_to_ARGBint(3MLIB),
mllib_VideoColorUYVY422int_to_ABGRint(3MLIB),
mllib_VideoColorUYV444int_to_ARGBint(3MLIB),
mllib_VideoColorUYV444int_to_ABGRint(3MLIB), attributes(5)

mllib_VideoColorYUV411seq_to_UYVY422int(3MLIB)

NAME	mllib_VideoColorYUV411seq_to_UYVY422int – convert YUV sequential to interleaved																		
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> void mllib_VideoColorYUV411seq_to_UYVY422int(mllib_u32 *UYVY, const mllib_u8 *Y, const mllib_u8 *U, const mllib_u8 *V, mllib_s32 <i>w</i>, mllib_s32 <i>h</i>, mllib_s32 <i>dlb</i>, mllib_s32 <i>ylb</i>, mllib_s32 <i>uvlb</i>);</pre>																		
DESCRIPTION	<p>The Y, U, V pixel streams are combined into a UYVY pixel stream. All pixel components are 8-bit unsigned integers. The Y buffer has dimensions <i>w</i> and <i>h</i>. The U and V buffers have dimensions <i>w</i>/4 and <i>h</i>. Dimension <i>w</i> is assumed to be a multiple of 4. In each row, every 4 Y values use the same U and V values.</p> <p>The following equation is used:</p> $\begin{aligned} \text{UYVY}[x][c/2] &= (\text{U}[r][c/4] \ll 24) \\ &(\text{Y}[r][c] \ll 16) \\ &(\text{V}[r][c/4] \ll 8) \\ &(\text{Y}[r][c+1]) \\ \text{UYVY}[x][c/2+1] &= (\text{U}[r][c/4] \ll 24) \\ &(\text{Y}[r][c+2] \ll 16) \\ &(\text{V}[r][c/4] \ll 8) \\ &(\text{Y}[r][c+3]) \end{aligned}$ <p>where $r = 0, 2, 4, \dots, h-2$; and $c = 0, 2, 4, \dots, w-2$.</p>																		
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>UYVY</i></td><td>Pointer to output buffer.</td></tr><tr><td><i>Y</i></td><td>Pointer to Y input buffer.</td></tr><tr><td><i>U</i></td><td>Pointer to U input buffer.</td></tr><tr><td><i>V</i></td><td>Pointer to V input buffer.</td></tr><tr><td><i>w</i></td><td>Image width in pixels.</td></tr><tr><td><i>h</i></td><td>Image height in lines.</td></tr><tr><td><i>dlb</i></td><td>Linebytes for UYVY buffer.</td></tr><tr><td><i>ylb</i></td><td>Linebytes for Y buffer.</td></tr><tr><td><i>uvlb</i></td><td>Linebytes for U and V buffers.</td></tr></table>	<i>UYVY</i>	Pointer to output buffer.	<i>Y</i>	Pointer to Y input buffer.	<i>U</i>	Pointer to U input buffer.	<i>V</i>	Pointer to V input buffer.	<i>w</i>	Image width in pixels.	<i>h</i>	Image height in lines.	<i>dlb</i>	Linebytes for UYVY buffer.	<i>ylb</i>	Linebytes for Y buffer.	<i>uvlb</i>	Linebytes for U and V buffers.
<i>UYVY</i>	Pointer to output buffer.																		
<i>Y</i>	Pointer to Y input buffer.																		
<i>U</i>	Pointer to U input buffer.																		
<i>V</i>	Pointer to V input buffer.																		
<i>w</i>	Image width in pixels.																		
<i>h</i>	Image height in lines.																		
<i>dlb</i>	Linebytes for UYVY buffer.																		
<i>ylb</i>	Linebytes for Y buffer.																		
<i>uvlb</i>	Linebytes for U and V buffers.																		
RETURN VALUES	None.																		
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:																		

mllib_VideoColorYUV411seq_to_UYVY422int(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

mllib_VideoColorYUV420seq_to_YUYV422int(3MLIB),
mllib_VideoColorYUV411seq_to_YUYV422int(3MLIB),
mllib_VideoColorYUV422seq_to_YUYV422int(3MLIB),
mllib_VideoColorYUV420seq_to_UYVY422int(3MLIB),
mllib_VideoColorYUV422seq_to_UYVY422int(3MLIB), attributes(5)

mllib_VideoColorYUV411seq_to_YUYV422int(3MLIB)

NAME	mllib_VideoColorYUV411seq_to_YUYV422int – convert YUV sequential to interleaved																		
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> void mllib_VideoColorYUV411seq_to_YUYV422int(mllib_u32 *YUYV, const mllib_u8 *Y, const mllib_u8 *U, const mllib_u8 *V, mllib_s32 <i>w</i>, mllib_s32 <i>h</i>, mllib_s32 <i>dlb</i>, mllib_s32 <i>ylb</i>, mllib_s32 <i>uvlb</i>);</pre>																		
DESCRIPTION	<p>The Y, U, V pixel streams are combined into a YUYV pixel stream. All pixel components are 8-bit unsigned integers. The Y buffer has dimensions <i>w</i> and <i>h</i>. The U and V buffers have dimensions <i>w</i>/4 and <i>h</i>. Dimension <i>w</i> is assumed to be a multiple of 4. In each row, every 4 Y values use the same U and V values.</p> <p>The following equation is used:</p> $\begin{aligned} \text{YUYV}[r][c/2] &= (\text{Y}[r][c] \ll 24) \\ &(\text{U}[r][c/4] \ll 16) \\ &(\text{Y}[r][c+1] \ll 8) \\ &(\text{V}[r][c/4]) \\ \text{YUYV}[r][c/2+1] &= (\text{Y}[r][c+2] \ll 24) \\ &(\text{U}[r][c/4] \ll 16) \\ &(\text{Y}[r][c+3] \ll 8) \\ &(\text{V}[r][c/4]) \end{aligned}$ <p>where $r = 0, 2, 4, \dots, h-2$; and $c = 0, 2, 4, \dots, w-2$.</p>																		
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>YUYV</i></td><td>Pointer to output buffer.</td></tr><tr><td><i>Y</i></td><td>Pointer to Y input buffer.</td></tr><tr><td><i>U</i></td><td>Pointer to U input buffer.</td></tr><tr><td><i>V</i></td><td>Pointer to V input buffer.</td></tr><tr><td><i>w</i></td><td>Image width in pixels.</td></tr><tr><td><i>h</i></td><td>Image height in lines.</td></tr><tr><td><i>dlb</i></td><td>Linebytes for YUYV buffer.</td></tr><tr><td><i>ylb</i></td><td>Linebytes for Y buffer.</td></tr><tr><td><i>uvlb</i></td><td>Linebytes for U and V buffers.</td></tr></table>	<i>YUYV</i>	Pointer to output buffer.	<i>Y</i>	Pointer to Y input buffer.	<i>U</i>	Pointer to U input buffer.	<i>V</i>	Pointer to V input buffer.	<i>w</i>	Image width in pixels.	<i>h</i>	Image height in lines.	<i>dlb</i>	Linebytes for YUYV buffer.	<i>ylb</i>	Linebytes for Y buffer.	<i>uvlb</i>	Linebytes for U and V buffers.
<i>YUYV</i>	Pointer to output buffer.																		
<i>Y</i>	Pointer to Y input buffer.																		
<i>U</i>	Pointer to U input buffer.																		
<i>V</i>	Pointer to V input buffer.																		
<i>w</i>	Image width in pixels.																		
<i>h</i>	Image height in lines.																		
<i>dlb</i>	Linebytes for YUYV buffer.																		
<i>ylb</i>	Linebytes for Y buffer.																		
<i>uvlb</i>	Linebytes for U and V buffers.																		
RETURN VALUES	None.																		
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:																		

mllib_VideoColorYUV411seq_to_YUYV422int(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

mllib_VideoColorYUV420seq_to_YUYV422int(3MLIB),
mllib_VideoColorYUV422seq_to_YUYV422int(3MLIB),
mllib_VideoColorYUV420seq_to_UYVY422int(3MLIB),
mllib_VideoColorYUV411seq_to_UYVY422int(3MLIB),
mllib_VideoColorYUV422seq_to_UYVY422int(3MLIB), attributes(5)

mllib_VideoColorYUV420seq_to_ABGRint(3MLIB)

NAME	mllib_VideoColorYUV420seq_to_ABGRint – color convert YUV sequential to ABGR interleaved																						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> void mllib_VideoColorYUV420seq_to_ABGRint(mllib_u32 *ABGR, const mllib_u8 *Y, const mllib_u8 *U, const mllib_u8 *V, const mllib_u8 *A_array, mllib_u8 A_const, mllib_s32 w, mllib_s32 h, mllib_s32 dlb, mllib_s32 yalb, mllib_s32 uvlb);</pre>																						
DESCRIPTION	<p>The Y, U, V pixel streams are converted into an ABGR pixel stream. All pixel components are 8-bit unsigned integers. The Y buffer has dimensions <i>w</i> and <i>h</i>. The U and V buffers have dimensions <i>w</i>/2 and <i>h</i>/2. Dimensions <i>w</i> and <i>h</i> are assumed to be even. Successive rows of the output buffer ABGR use successive rows of Y and the same rows of U and V.</p> <p>The alpha values for this function work in the following fashion:</p> <ul style="list-style-type: none"> ■ If <i>A_array</i> pointer is not NULL, the values are taken from there. It has to have the same dimensions as the Y buffer. ■ If <i>A_array</i> pointer is NULL, the alpha values for every pixel are set to <i>A_const</i>. <p>The following equation is used:</p> $\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.1644 & 0.0000 & 1.5966 \\ 1.1644 & -0.3920 & -0.8132 \\ 1.1644 & 2.0184 & 0.0000 \end{bmatrix} * \begin{bmatrix} Y \\ U \\ V \end{bmatrix} - \begin{bmatrix} 16.0000 \\ 128.0000 \\ 128.0000 \end{bmatrix}$																						
PARAMETERS	<p>The function takes the following arguments:</p> <table border="0" style="width: 100%;"> <tr> <td style="padding-right: 20px;"><i>ABGR</i></td> <td>Pointer to output buffer.</td> </tr> <tr> <td><i>Y</i></td> <td>Pointer to Y input buffer.</td> </tr> <tr> <td><i>U</i></td> <td>Pointer to U input buffer.</td> </tr> <tr> <td><i>V</i></td> <td>Pointer to V input buffer.</td> </tr> <tr> <td><i>A_array</i></td> <td>Array of alpha values.</td> </tr> <tr> <td><i>A_const</i></td> <td>Constant alpha value.</td> </tr> <tr> <td><i>w</i></td> <td>Image width in pixels.</td> </tr> <tr> <td><i>h</i></td> <td>Image height in lines.</td> </tr> <tr> <td><i>dlb</i></td> <td>Linebytes for output buffer.</td> </tr> <tr> <td><i>yalb</i></td> <td>Linebytes for Y and alpha buffers.</td> </tr> <tr> <td><i>uvlb</i></td> <td>Linebytes for U and V buffers.</td> </tr> </table>	<i>ABGR</i>	Pointer to output buffer.	<i>Y</i>	Pointer to Y input buffer.	<i>U</i>	Pointer to U input buffer.	<i>V</i>	Pointer to V input buffer.	<i>A_array</i>	Array of alpha values.	<i>A_const</i>	Constant alpha value.	<i>w</i>	Image width in pixels.	<i>h</i>	Image height in lines.	<i>dlb</i>	Linebytes for output buffer.	<i>yalb</i>	Linebytes for Y and alpha buffers.	<i>uvlb</i>	Linebytes for U and V buffers.
<i>ABGR</i>	Pointer to output buffer.																						
<i>Y</i>	Pointer to Y input buffer.																						
<i>U</i>	Pointer to U input buffer.																						
<i>V</i>	Pointer to V input buffer.																						
<i>A_array</i>	Array of alpha values.																						
<i>A_const</i>	Constant alpha value.																						
<i>w</i>	Image width in pixels.																						
<i>h</i>	Image height in lines.																						
<i>dlb</i>	Linebytes for output buffer.																						
<i>yalb</i>	Linebytes for Y and alpha buffers.																						
<i>uvlb</i>	Linebytes for U and V buffers.																						
RETURN VALUES	None.																						

mllib_VideoColorYUV420seq_to_ABGRint(3MLIB)

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorYUV420seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV411seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV422seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV444seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV411seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV422seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV444seq_to_ABGRint(3MLIB),
mllib_VideoColorYUYV422int_to_ARGBint(3MLIB),
mllib_VideoColorYUV444int_to_ARGBint(3MLIB),
mllib_VideoColorYUYV422int_to_ABGRint(3MLIB),
mllib_VideoColorYUV444int_to_ABGRint(3MLIB),
mllib_VideoColorUYVY422int_to_ARGBint(3MLIB),
mllib_VideoColorUYVY422int_to_ABGRint(3MLIB),
mllib_VideoColorUYV444int_to_ARGBint(3MLIB),
mllib_VideoColorUYV444int_to_ABGRint(3MLIB), attributes(5)

mllib_VideoColorYUV420seq_to_ARGBint(3MLIB)

NAME	mllib_VideoColorYUV420seq_to_ARGBint – color convert YUV sequential to ARGB interleaved																						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> void mllib_VideoColorYUV420seq_to_ARGBint(mllib_u32 *ARGB, const mllib_u8 *Y, const mllib_u8 *U, const mllib_u8 *V, const mllib_u8 *A_array, mllib_u8 A_const, mllib_s32 w, mllib_s32 h, mllib_s32 dlb, mllib_s32 yalb, mllib_s32 uvlb);</pre>																						
DESCRIPTION	<p>The Y, U, V pixel streams are converted into an ARGB pixel stream. All pixel components are 8-bit unsigned integers. The Y buffer has dimensions <i>w</i> and <i>h</i>. The U and V buffers have dimensions <i>w</i>/2 and <i>h</i>/2. Dimensions <i>w</i> and <i>h</i> are assumed to be even. Successive rows of the output buffer ARGB use successive rows of Y and the same rows of U and V.</p> <p>The alpha values for this function work in the following fashion:</p> <ul style="list-style-type: none">■ If <i>A_array</i> pointer is not NULL, the values are taken from there. It has to have the same dimensions as the Y buffer.■ If <i>A_array</i> pointer is NULL, the alpha values for every pixel are set to <i>A_const</i>. <p>The following equation is used:</p> $\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.1644 & 0.0000 & 1.5966 \\ 1.1644 & -0.3920 & -0.8132 \\ 1.1644 & 2.0184 & 0.0000 \end{bmatrix} * \begin{bmatrix} Y \\ U \\ V \end{bmatrix} - \begin{bmatrix} 16.0000 \\ 128.0000 \\ 128.0000 \end{bmatrix}$																						
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>ARGB</i></td><td>Pointer to output buffer.</td></tr><tr><td><i>Y</i></td><td>Pointer to Y input buffer.</td></tr><tr><td><i>U</i></td><td>Pointer to U input buffer.</td></tr><tr><td><i>V</i></td><td>Pointer to V input buffer.</td></tr><tr><td><i>A_array</i></td><td>Array of alpha values.</td></tr><tr><td><i>A_const</i></td><td>Constant alpha value.</td></tr><tr><td><i>w</i></td><td>Image width in pixels.</td></tr><tr><td><i>h</i></td><td>Image height in lines.</td></tr><tr><td><i>dlb</i></td><td>Linebytes for output buffer.</td></tr><tr><td><i>yalb</i></td><td>Linebytes for Y and alpha buffers.</td></tr><tr><td><i>uvlb</i></td><td>Linebytes for U and V buffers.</td></tr></table>	<i>ARGB</i>	Pointer to output buffer.	<i>Y</i>	Pointer to Y input buffer.	<i>U</i>	Pointer to U input buffer.	<i>V</i>	Pointer to V input buffer.	<i>A_array</i>	Array of alpha values.	<i>A_const</i>	Constant alpha value.	<i>w</i>	Image width in pixels.	<i>h</i>	Image height in lines.	<i>dlb</i>	Linebytes for output buffer.	<i>yalb</i>	Linebytes for Y and alpha buffers.	<i>uvlb</i>	Linebytes for U and V buffers.
<i>ARGB</i>	Pointer to output buffer.																						
<i>Y</i>	Pointer to Y input buffer.																						
<i>U</i>	Pointer to U input buffer.																						
<i>V</i>	Pointer to V input buffer.																						
<i>A_array</i>	Array of alpha values.																						
<i>A_const</i>	Constant alpha value.																						
<i>w</i>	Image width in pixels.																						
<i>h</i>	Image height in lines.																						
<i>dlb</i>	Linebytes for output buffer.																						
<i>yalb</i>	Linebytes for Y and alpha buffers.																						
<i>uvlb</i>	Linebytes for U and V buffers.																						
RETURN VALUES	None.																						

mllib_VideoColorYUV420seq_to_ARGBint(3MLIB)

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorYUV411seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV422seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV444seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV420seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV411seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV422seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV444seq_to_ABGRint(3MLIB),
mllib_VideoColorYUYV422int_to_ARGBint(3MLIB),
mllib_VideoColorYUV444int_to_ARGBint(3MLIB),
mllib_VideoColorYUYV422int_to_ABGRint(3MLIB),
mllib_VideoColorYUV444int_to_ABGRint(3MLIB),
mllib_VideoColorUYVY422int_to_ARGBint(3MLIB),
mllib_VideoColorUYVY422int_to_ABGRint(3MLIB),
mllib_VideoColorUYV444int_to_ARGBint(3MLIB),
mllib_VideoColorUYV444int_to_ABGRint(3MLIB), attributes(5)

mllib_VideoColorYUV420seq_to_UYVY422int(3MLIB)

NAME	mllib_VideoColorYUV420seq_to_UYVY422int – convert YUV sequential to interleaved																		
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> void mllib_VideoColorYUV420seq_to_UYVY422int(mllib_u32 *UYVY, const mllib_u8 *Y, const mllib_u8 *U, const mllib_u8 *V, mllib_s32 w, mllib_s32 h, mllib_s32 dlb, mllib_s32 ylb, mllib_s32 uvlb);</pre>																		
DESCRIPTION	<p>The Y, U, V pixel streams are combined into a UYVY pixel stream. All pixel components are 8-bit unsigned integers. The Y buffer has dimensions <i>w</i> and <i>h</i>. The U and V buffers have dimensions <i>w</i>/2 and <i>h</i>/2. Dimensions <i>w</i> and <i>h</i> are assumed to be even. Successive rows of the output buffer UYVY use successive rows of Y and the same rows of U and V.</p> <p>The following equation is used:</p> $\begin{aligned} \text{UYVY}[r][c/2] &= (\text{U}[r/2][c/2] \ll 24) \\ &(\text{Y}[r][c] \ll 16) \\ &(\text{V}[r/2][c/2] \ll 8) \\ &(\text{Y}[r][c+1]) \\ \\ \text{UYVY}[r+1][c/2] &= (\text{U}[r/2][c/2] \ll 24) \\ &(\text{Y}[r+1][c] \ll 16) \\ &(\text{V}[r/2][c/2] \ll 8) \\ &(\text{Y}[r+1][c+1]) \end{aligned}$ <p>where $r = 0, 2, 4, \dots, h-2$; and $c = 0, 2, 4, \dots, w-2$.</p>																		
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>UYVY</i></td><td>Pointer to output buffer.</td></tr><tr><td><i>Y</i></td><td>Pointer to Y input buffer.</td></tr><tr><td><i>U</i></td><td>Pointer to U input buffer.</td></tr><tr><td><i>V</i></td><td>Pointer to V input buffer.</td></tr><tr><td><i>w</i></td><td>Image width in pixels.</td></tr><tr><td><i>h</i></td><td>Image height in lines.</td></tr><tr><td><i>dlb</i></td><td>Linebytes for UYVY buffer.</td></tr><tr><td><i>ylb</i></td><td>Linebytes for Y buffer.</td></tr><tr><td><i>uvlb</i></td><td>Linebytes for U and V buffers.</td></tr></table>	<i>UYVY</i>	Pointer to output buffer.	<i>Y</i>	Pointer to Y input buffer.	<i>U</i>	Pointer to U input buffer.	<i>V</i>	Pointer to V input buffer.	<i>w</i>	Image width in pixels.	<i>h</i>	Image height in lines.	<i>dlb</i>	Linebytes for UYVY buffer.	<i>ylb</i>	Linebytes for Y buffer.	<i>uvlb</i>	Linebytes for U and V buffers.
<i>UYVY</i>	Pointer to output buffer.																		
<i>Y</i>	Pointer to Y input buffer.																		
<i>U</i>	Pointer to U input buffer.																		
<i>V</i>	Pointer to V input buffer.																		
<i>w</i>	Image width in pixels.																		
<i>h</i>	Image height in lines.																		
<i>dlb</i>	Linebytes for UYVY buffer.																		
<i>ylb</i>	Linebytes for Y buffer.																		
<i>uvlb</i>	Linebytes for U and V buffers.																		
RETURN VALUES	None.																		
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:																		

mllib_VideoColorYUV420seq_to_UYVY422int(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

mllib_VideoColorYUV420seq_to_YUYV422int(3MLIB),
mllib_VideoColorYUV411seq_to_YUYV422int(3MLIB),
mllib_VideoColorYUV422seq_to_YUYV422int(3MLIB),
mllib_VideoColorYUV411seq_to_UYVY422int(3MLIB),
mllib_VideoColorYUV422seq_to_UYVY422int(3MLIB), attributes(5)

mllib_VideoColorYUV420seq_to_YUYV422int(3MLIB)

NAME	mllib_VideoColorYUV420seq_to_YUYV422int – convert YUV sequential to interleaved
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> void mllib_VideoColorYUV420seq_to_YUYV422int(mllib_u32 *YUYV, const mllib_u8 *Y, const mllib_u8 *U, const mllib_u8 *V, mllib_s32 w, mllib_s32 h, mllib_s32 dlb, mllib_s32 ylb, mllib_s32 uvlb);</pre>
DESCRIPTION	<p>The Y, U, V pixel streams are combined into a YUYV pixel stream. All pixel components are 8-bit unsigned integers. The Y buffer has dimensions w and h. The U and V buffers have dimensions $w/2$ and $h/2$. Dimensions w and h are assumed to be even. Successive rows of the output buffer YUYV use successive rows of Y and the same rows of U and V.</p> <p>The following equation is used:</p> $\begin{aligned} \text{YUYV}[r][c/2] &= (\text{Y}[r][c] \ll 24) \\ &(\text{U}[r/2][c/2] \ll 16) \\ &(\text{Y}[r][c+1] \ll 8) \\ &(\text{V}[r/2][c/2]) \end{aligned}$ $\begin{aligned} \text{YUYV}[r+1][c/2] &= (\text{Y}[r+1][c] \ll 24) \\ &(\text{U}[r/2][c/2] \ll 16) \\ &(\text{Y}[r+1][c+1] \ll 8) \\ &(\text{V}[r/2][c/2]) \end{aligned}$ <p>where $r = 0, 2, 4, \dots, h-2$; and $c = 0, 2, 4, \dots, w-2$.</p>
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>YUYV</i> Pointer to output buffer.</p> <p><i>Y</i> Pointer to Y input buffer.</p> <p><i>U</i> Pointer to U input buffer.</p> <p><i>V</i> Pointer to V input buffer.</p> <p><i>w</i> Image width in pixels.</p> <p><i>h</i> Image height in lines.</p> <p><i>dlb</i> Linebytes for YUYV buffer.</p> <p><i>ylb</i> Linebytes for Y buffer.</p> <p><i>uvlb</i> Linebytes for U and V buffers.</p>
RETURN VALUES	None.
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:

mllib_VideoColorYUV420seq_to_YUYV422int(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

mllib_VideoColorYUV411seq_to_YUYV422int(3MLIB),
mllib_VideoColorYUV422seq_to_YUYV422int(3MLIB),
mllib_VideoColorYUV420seq_to_UYVY422int(3MLIB),
mllib_VideoColorYUV411seq_to_UYVY422int(3MLIB),
mllib_VideoColorYUV422seq_to_UYVY422int(3MLIB), attributes(5)

mllib_VideoColorYUV422seq_to_ABGRint(3MLIB)

NAME	mllib_VideoColorYUV422seq_to_ABGRint – color convert YUV sequential to ABGR interleaved																						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> void mllib_VideoColorYUV422seq_to_ABGRint(mllib_u32 *ABGR, const mllib_u8 *Y, const mllib_u8 *U, const mllib_u8 *V, const mllib_u8 *A_array, mllib_u8 A_const, mllib_s32 w, mllib_s32 h, mllib_s32 dlb, mllib_s32 yalb, mllib_s32 uvlb);</pre>																						
DESCRIPTION	<p>The Y, U, V pixel streams are converted into an ABGR pixel stream. All pixel components are 8-bit unsigned integers. The Y buffer has dimensions <i>w</i> and <i>h</i>. The U and V buffers have dimensions <i>w</i>/2 and <i>h</i>. Dimensions <i>w</i> and <i>h</i> are assumed to be even. Similar to mllib_VideoColorYUV420seq_to_ABGRint() except U and V are not sampled in the <i>h</i> direction.</p> <p>The alpha values for this function work in the following fashion:</p> <ul style="list-style-type: none"> ■ If <i>A_array</i> pointer is not NULL, the values are taken from there. It has to have the same dimensions as the Y buffer. ■ If <i>A_array</i> pointer is NULL, the alpha values for every pixel are set to <i>A_const</i>. <p>The following equation is used:</p> $\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.1644 & 0.0000 & 1.5966 \\ 1.1644 & -0.3920 & -0.8132 \\ 1.1644 & 2.0184 & 0.0000 \end{bmatrix} * \begin{bmatrix} Y \\ U \\ V \end{bmatrix} - \begin{bmatrix} 16.0000 \\ 128.0000 \\ 128.0000 \end{bmatrix}$																						
PARAMETERS	<p>The function takes the following arguments:</p> <table border="0" style="width: 100%;"> <tr> <td style="padding-right: 20px;"><i>ABGR</i></td> <td>Pointer to output buffer.</td> </tr> <tr> <td><i>Y</i></td> <td>Pointer to Y input buffer.</td> </tr> <tr> <td><i>U</i></td> <td>Pointer to U input buffer.</td> </tr> <tr> <td><i>V</i></td> <td>Pointer to V input buffer.</td> </tr> <tr> <td><i>A_array</i></td> <td>Array of alpha values.</td> </tr> <tr> <td><i>A_const</i></td> <td>Constant alpha value.</td> </tr> <tr> <td><i>w</i></td> <td>Image width in pixels.</td> </tr> <tr> <td><i>h</i></td> <td>Image height in lines.</td> </tr> <tr> <td><i>dlb</i></td> <td>Linebytes for output buffer.</td> </tr> <tr> <td><i>yalb</i></td> <td>Linebytes for Y and alpha buffers.</td> </tr> <tr> <td><i>uvlb</i></td> <td>Linebytes for U and V buffers.</td> </tr> </table>	<i>ABGR</i>	Pointer to output buffer.	<i>Y</i>	Pointer to Y input buffer.	<i>U</i>	Pointer to U input buffer.	<i>V</i>	Pointer to V input buffer.	<i>A_array</i>	Array of alpha values.	<i>A_const</i>	Constant alpha value.	<i>w</i>	Image width in pixels.	<i>h</i>	Image height in lines.	<i>dlb</i>	Linebytes for output buffer.	<i>yalb</i>	Linebytes for Y and alpha buffers.	<i>uvlb</i>	Linebytes for U and V buffers.
<i>ABGR</i>	Pointer to output buffer.																						
<i>Y</i>	Pointer to Y input buffer.																						
<i>U</i>	Pointer to U input buffer.																						
<i>V</i>	Pointer to V input buffer.																						
<i>A_array</i>	Array of alpha values.																						
<i>A_const</i>	Constant alpha value.																						
<i>w</i>	Image width in pixels.																						
<i>h</i>	Image height in lines.																						
<i>dlb</i>	Linebytes for output buffer.																						
<i>yalb</i>	Linebytes for Y and alpha buffers.																						
<i>uvlb</i>	Linebytes for U and V buffers.																						
RETURN VALUES	None.																						

mllib_VideoColorYUV422seq_to_ABGRint(3MLIB)

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorYUV420seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV411seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV422seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV444seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV420seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV411seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV444seq_to_ABGRint(3MLIB),
mllib_VideoColorYUYV422int_to_ARGBint(3MLIB),
mllib_VideoColorYUV444int_to_ARGBint(3MLIB),
mllib_VideoColorYUYV422int_to_ABGRint(3MLIB),
mllib_VideoColorYUV444int_to_ABGRint(3MLIB),
mllib_VideoColorUYVY422int_to_ARGBint(3MLIB),
mllib_VideoColorUYVY422int_to_ABGRint(3MLIB),
mllib_VideoColorUYV444int_to_ARGBint(3MLIB),
mllib_VideoColorUYV444int_to_ABGRint(3MLIB), attributes(5)

mllib_VideoColorYUV422seq_to_ARGBint(3MLIB)

NAME	mllib_VideoColorYUV422seq_to_ARGBint – color convert YUV sequential to ARGB interleaved																						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> void mllib_VideoColorYUV422seq_to_ARGBint(mllib_u32 *ARGB, const mllib_u8 *Y, const mllib_u8 *U, const mllib_u8 *V, const mllib_u8 *A_array, mllib_u8 A_const, mllib_s32 w, mllib_s32 h, mllib_s32 dlb, mllib_s32 yalb, mllib_s32 uvlb);</pre>																						
DESCRIPTION	<p>The Y, U, V pixel streams are converted into an ARGB pixel stream. All pixel components are 8-bit unsigned integers. The Y buffer has dimensions <i>w</i> and <i>h</i>. The U and V buffers have dimensions <i>w</i>/2 and <i>h</i>. Dimensions <i>w</i> and <i>h</i> are assumed to be even. Similar to mllib_VideoColorYUV420seq_to_ARGBint() except U and V are not sampled in the <i>h</i> direction.</p> <p>The alpha values for this function work in the following fashion:</p> <ul style="list-style-type: none">■ If <i>A_array</i> pointer is not NULL, the values are taken from there. It has to have the same dimensions as the Y buffer.■ If <i>A_array</i> pointer is NULL, the alpha values for every pixel are set to <i>A_const</i>. <p>The following equation is used:</p> $\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.1644 & 0.0000 & 1.5966 \\ 1.1644 & -0.3920 & -0.8132 \\ 1.1644 & 2.0184 & 0.0000 \end{bmatrix} * \begin{bmatrix} Y \\ U \\ V \end{bmatrix} - \begin{bmatrix} 16.0000 \\ 128.0000 \\ 128.0000 \end{bmatrix}$																						
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>ARGB</i></td><td>Pointer to output buffer.</td></tr><tr><td><i>Y</i></td><td>Pointer to Y input buffer.</td></tr><tr><td><i>U</i></td><td>Pointer to U input buffer.</td></tr><tr><td><i>V</i></td><td>Pointer to V input buffer.</td></tr><tr><td><i>A_array</i></td><td>Array of alpha values.</td></tr><tr><td><i>A_const</i></td><td>Constant alpha value.</td></tr><tr><td><i>w</i></td><td>Image width in pixels.</td></tr><tr><td><i>h</i></td><td>Image height in lines.</td></tr><tr><td><i>dlb</i></td><td>Linebytes for output buffer.</td></tr><tr><td><i>yalb</i></td><td>Linebytes for Y and alpha buffers.</td></tr><tr><td><i>uvlb</i></td><td>Linebytes for U and V buffers.</td></tr></table>	<i>ARGB</i>	Pointer to output buffer.	<i>Y</i>	Pointer to Y input buffer.	<i>U</i>	Pointer to U input buffer.	<i>V</i>	Pointer to V input buffer.	<i>A_array</i>	Array of alpha values.	<i>A_const</i>	Constant alpha value.	<i>w</i>	Image width in pixels.	<i>h</i>	Image height in lines.	<i>dlb</i>	Linebytes for output buffer.	<i>yalb</i>	Linebytes for Y and alpha buffers.	<i>uvlb</i>	Linebytes for U and V buffers.
<i>ARGB</i>	Pointer to output buffer.																						
<i>Y</i>	Pointer to Y input buffer.																						
<i>U</i>	Pointer to U input buffer.																						
<i>V</i>	Pointer to V input buffer.																						
<i>A_array</i>	Array of alpha values.																						
<i>A_const</i>	Constant alpha value.																						
<i>w</i>	Image width in pixels.																						
<i>h</i>	Image height in lines.																						
<i>dlb</i>	Linebytes for output buffer.																						
<i>yalb</i>	Linebytes for Y and alpha buffers.																						
<i>uvlb</i>	Linebytes for U and V buffers.																						
RETURN VALUES	None.																						

mllib_VideoColorYUV422seq_to_ARGBint(3MLIB)

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoColorYUV420seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV411seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV444seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV420seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV411seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV422seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV444seq_to_ABGRint(3MLIB),
mllib_VideoColorYUYV422int_to_ARGBint(3MLIB),
mllib_VideoColorYUV444int_to_ARGBint(3MLIB),
mllib_VideoColorYUYV422int_to_ABGRint(3MLIB),
mllib_VideoColorYUV444int_to_ABGRint(3MLIB),
mllib_VideoColorUYVY422int_to_ARGBint(3MLIB),
mllib_VideoColorUYVY422int_to_ABGRint(3MLIB),
mllib_VideoColorUYV444int_to_ARGBint(3MLIB),
mllib_VideoColorUYV444int_to_ABGRint(3MLIB), attributes(5)

mllib_VideoColorYUV422seq_to_UYVY422int(3MLIB)

NAME mllib_VideoColorYUV422seq_to_UYVY422int – convert YUV sequential to interleaved

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_VideoColorYUV422seq_to_UYVY422int(mllib_u32 *UYVY,
      const mllib_u8 *Y, const mllib_u8 *U, const mllib_u8 *V,
      mllib_s32 w, mllib_s32 h, mllib_s32 dlb, mllib_s32 ylb, mllib_s32
      uvlb);
```

DESCRIPTION

The Y, U, V pixel streams are combined into a UYVY pixel stream. All pixel components are 8-bit unsigned integers. The Y buffer has dimensions *w* and *h*. The U and V buffers have dimensions *w*/2 and *h*. Dimensions *w* and *h* are assumed to be even. Similar to `mllib_VideoColorYUV420seq_to_UYVY422int()` except U and V are not sampled in the *h* direction.

The following equation is used:

$$UYVY[r][c/2] = (U[r][c/2] \ll 24) | (Y[r][c] \ll 16) | (V[r][c/2] \ll 8) | (Y[r][c+1])$$

where $r = 0, 1, 2, \dots, h-1$; and $c = 0, 2, 4, \dots, w-2$.

PARAMETERS

The function takes the following arguments:

UYVY Pointer to output buffer.

Y Pointer to Y input buffer.

U Pointer to U input buffer.

V Pointer to V input buffer.

w Image width in pixels.

h Image height in lines.

dlb Linebytes for UYVY buffer.

ylb Linebytes for Y buffer.

uvlb Linebytes for U and V buffers.

RETURN VALUES None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

`mlib_VideoColorYUV422seq_to_UYVY422int(3MLIB)`

SEE ALSO | `mlib_VideoColorYUV420seq_to_YUYV422int(3MLIB)`,
`mlib_VideoColorYUV411seq_to_YUYV422int(3MLIB)`,
`mlib_VideoColorYUV422seq_to_YUYV422int(3MLIB)`,
`mlib_VideoColorYUV420seq_to_UYVY422int(3MLIB)`,
`mlib_VideoColorYUV411seq_to_UYVY422int(3MLIB)`, `attributes(5)`

mllib_VideoColorYUV422seq_to_YUYV422int(3MLIB)

NAME mllib_VideoColorYUV422seq_to_YUYV422int – convert YUV sequential to interleaved

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_VideoColorYUV422seq_to_YUYV422int(mllib_u32 *YUYV,
      const mllib_u8 *Y, const mllib_u8 *U, const mllib_u8 *V,
      mllib_s32 w, mllib_s32 h, mllib_s32 dlb, mllib_s32 ylb, mllib_s32
      uvlb);
```

DESCRIPTION

The Y, U, V pixel streams are combined into a YUYV pixel stream. All pixel components are 8-bit unsigned integers. The Y buffer has dimensions *w* and *h*. The U and V buffers have dimensions *w*/2 and *h*. Dimensions *w* and *h* are assumed to be even. Similar to `mllib_VideoColorYUV420seq_to_YUYV422int()` except U and V are not sampled in the *h* direction.

The following equation is used:

$$YUYV[r][c/2] = (Y[r][c] \ll 24) | (U[r][c/2] \ll 16) | (Y[r][c+1] \ll 8) | (V[r][c/2])$$

where $r = 0, 1, 2, \dots, h-1$; and $c = 0, 2, 4, \dots, w-2$.

PARAMETERS

The function takes the following arguments:

YUYV Pointer to output buffer.

Y Pointer to Y input buffer.

U Pointer to U input buffer.

V Pointer to V input buffer.

w Image width in pixels.

h Image height in lines.

dlb Linebytes for YUYV buffer.

ylb Linebytes for Y buffer.

uvlb Linebytes for U and V buffers.

RETURN VALUES None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

`mlib_VideoColorYUV422seq_to_YUYV422int(3MLIB)`

SEE ALSO | `mlib_VideoColorYUV420seq_to_YUYV422int(3MLIB)`,
`mlib_VideoColorYUV411seq_to_YUYV422int(3MLIB)`,
`mlib_VideoColorYUV420seq_to_UYVY422int(3MLIB)`,
`mlib_VideoColorYUV411seq_to_UYVY422int(3MLIB)`,
`mlib_VideoColorYUV422seq_to_UYVY422int(3MLIB)`, `attributes(5)`

mllib_VideoColorYUV444int_to_ABGRint(3MLIB)

NAME mllib_VideoColorYUV444int_to_ABGRint – color convert YUV interleaved to ABGR interleaved

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_VideoColorYUV444int_to_ABGRint(mllib_u32 *ABGR, const
    mllib_u8 *YUV, const mllib_u8 *A_array, mllib_u8 A_const, mllib_s32
    w, mllib_s32 h, mllib_s32 dlb, mllib_s32 slb, mllib_s32 alb);
```

DESCRIPTION The YUV pixel stream is converted into an ABGR pixel stream. All pixel components are 8-bit unsigned integers. All buffers have dimensions *w* and *h*.

The alpha values for this function work in the following fashion:

- If *A_array* pointer is not NULL, the values are taken from there. It has to have the same dimensions as the Y buffer.
- If *A_array* pointer is NULL, the alpha values for every pixel are set to *A_const*.

The following equation is used:

$$\begin{bmatrix} |R| \\ |G| \\ |B| \end{bmatrix} = \begin{bmatrix} |1.1644 & 0.0000 & 1.5966| \\ |1.1644 & -0.3920 & -0.8132| \\ |1.1644 & 2.0184 & 0.0000| \end{bmatrix} * \begin{bmatrix} [|Y| \\ |U| \\ |V| \end{bmatrix} - \begin{bmatrix} |16.0000| \\ |128.0000| \\ |128.0000| \end{bmatrix}$$

PARAMETERS The function takes the following arguments:

ABGR Pointer to output buffer.
YUV Pointer to Y input buffer.
A_array Array of alpha values.
A_const Constant alpha value.
w Image width in pixels.
h Image height in lines.
dlb Linebytes for output buffer.
slb Linebytes for input buffer.
alb Linebytes for alpha buffer.

RETURN VALUES None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

mllib_VideoColorYUV444int_to_ABGRint(3MLIB)

SEE ALSO | mllib_VideoColorYUV420seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV411seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV422seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV444seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV420seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV411seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV422seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV444seq_to_ABGRint(3MLIB),
mllib_VideoColorYUYV422int_to_ARGBint(3MLIB),
mllib_VideoColorYUV444int_to_ARGBint(3MLIB),
mllib_VideoColorYUYV422int_to_ABGRint(3MLIB),
mllib_VideoColorUYVY422int_to_ARGBint(3MLIB),
mllib_VideoColorUYVY422int_to_ABGRint(3MLIB),
mllib_VideoColorUYV444int_to_ARGBint(3MLIB),
mllib_VideoColorUYV444int_to_ABGRint(3MLIB), attributes(5)

mllib_VideoColorYUV444int_to_ARGBint(3MLIB)

NAME mllib_VideoColorYUV444int_to_ARGBint – color convert YUV interleaved to ARGB interleaved

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_VideoColorYUV444int_to_ARGBint(mllib_u32 *ARGB, const
    mllib_u8 *YUV, const mllib_u8 *A_array, mllib_u8 A_const, mllib_s32
    w, mllib_s32 h, mllib_s32 dlb, mllib_s32 slb, mllib_s32 alb);
```

DESCRIPTION

The YUV pixel stream is converted into an ARGB pixel stream. All pixel components are 8-bit unsigned integers. All buffers have dimensions *w* and *h*.

The alpha values for this function work in the following fashion:

- If *A_array* pointer is not NULL, the values are taken from there. It has to have the same dimensions as the Y buffer.
- If *A_array* pointer is NULL, the alpha values for every pixel are set to *A_const*.

The following equation is used:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.1644 & 0.0000 & 1.5966 \\ 1.1644 & -0.3920 & -0.8132 \\ 1.1644 & 2.0184 & 0.0000 \end{bmatrix} * \begin{bmatrix} Y \\ U \\ V \end{bmatrix} - \begin{bmatrix} 16.0000 \\ 128.0000 \\ 128.0000 \end{bmatrix}$$

PARAMETERS

The function takes the following arguments:

ARGB Pointer to output buffer.

YUV Pointer to Y input buffer.

A_array Array of alpha values.

A_const Constant alpha value.

w Image width in pixels.

h Image height in lines.

dlb Linebytes for output buffer.

slb Linebytes for input buffer.

alb Linebytes for alpha buffer.

RETURN VALUES None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

mllib_VideoColorYUV444int_to_ARGBint(3MLIB)

SEE ALSO | mllib_VideoColorYUV420seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV411seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV422seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV444seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV420seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV411seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV422seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV444seq_to_ABGRint(3MLIB),
mllib_VideoColorYUYV422int_to_ARGBint(3MLIB),
mllib_VideoColorYUYV422int_to_ABGRint(3MLIB),
mllib_VideoColorYUV444int_to_ABGRint(3MLIB),
mllib_VideoColorUYVY422int_to_ARGBint(3MLIB),
mllib_VideoColorUYVY422int_to_ABGRint(3MLIB),
mllib_VideoColorUYV444int_to_ARGBint(3MLIB),
mllib_VideoColorUYV444int_to_ABGRint(3MLIB), attributes(5)

mllib_VideoColorYUV444int_to_UYVY422int(3MLIB)

NAME mllib_VideoColorYUV444int_to_UYVY422int – convert YUV interleaved with subsampling

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_VideoColorYUV444int_to_UYVY422int(mllib_u32 *UYVY,
      const mllib_u8 *YUV, mllib_s32 w, mllib_s32 h, mllib_s32 dlb,
      mllib_s32 slb);
```

DESCRIPTION The YUV pixel stream is broken apart and recombined into a UYVY pixel stream. All pixel components are 8-bit unsigned integers. The YUV buffer has dimensions *w* and *h*. Dimension *w* is assumed to be a multiple of 2. Adjacent U and V values are averaged to get the output U and V values. The sequence of values in the input stream is Y[r][c], U[r][c], V[r][c], Y[r][c+1], U[r][c+1], V[r][c+1], ...

The following equation is used:

$$UYVY[r][c/2] = (((U[r][c] + U[r][c+1]) / 2) \ll 24) | (Y[r][c] \ll 16) | (((V[r][c] + V[r][c+1]) / 2) \ll 8) | (Y[r][c+1])$$

where *r* = 0, 1, 2, ..., *h*-1; and *c* = 0, 2, 4, ..., *w*-2.

PARAMETERS The function takes the following arguments:

UYVY Pointer to output buffer.

YUV Pointer to input buffer.

w Image width in pixels.

h Image height in lines.

dlb Linebytes for output buffer.

slb Linebytes for input buffer.

RETURN VALUES None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO [mllib_VideoColorYUV444seq_to_YUYV422int\(3MLIB\)](#), [mllib_VideoColorYUV444int_to_YUYV422int\(3MLIB\)](#), [mllib_VideoColorYUV444seq_to_UYVY422int\(3MLIB\)](#), [mllib_VideoColorUYV444int_to_YUYV422int\(3MLIB\)](#), [mllib_VideoColorUYV444int_to_UYVY422int\(3MLIB\)](#), [attributes\(5\)](#)

mllib_VideoColorYUV444int_to_YUYV422int(3MLIB)

NAME | mllib_VideoColorYUV444int_to_YUYV422int – convert YUV interleaved with subsampling

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_VideoColorYUV444int_to_YUYV422int(mllib_u32 *YUYV,
      const mllib_u8 *YUV, mllib_s32 w, mllib_s32 h, mllib_s32 dlb,
      mllib_s32 slb);
```

DESCRIPTION | The YUV pixel stream is broken apart and recombined into a YUYV pixel stream. All pixel components are 8-bit unsigned integers. The YUV buffer has dimensions *w* and *h*. Dimension *w* is assumed to be a multiple of 2. Adjacent U and V values are averaged to get the output U and V values. The sequence of values in the input stream is Y[r][c], U[r][c], V[r][c], Y[r][c+1], U[r][c+1], V[r][c+1], ...

The following equation is used:

$$YUYV[r][c/2] = (Y[r][c] \ll 24) | ((U[r][c] + U[r][c+1]) / 2) \ll 16 | (Y[r][c+1] \ll 8) | ((V[r][c] + V[r][c+1]) / 2)$$

where *r* = 0, 1, 2, ..., *h*-1; and *c* = 0, 2, 4, ..., *w*-2.

PARAMETERS | The function takes the following arguments:

YUYV | Pointer to output buffer.

YUV | Pointer to input buffer.

w | Image width in pixels.

h | Image height in lines.

dlb | Linebytes for output buffer.

slb | Linebytes for input buffers.

RETURN VALUES | None.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_VideoColorYUV444seq_to_YUYV422int(3MLIB), mllib_VideoColorYUV444seq_to_UYVY422int(3MLIB), mllib_VideoColorYUV444int_to_UYVY422int(3MLIB), mllib_VideoColorUYV444int_to_YUYV422int(3MLIB), mllib_VideoColorUYV444int_to_UYVY422int(3MLIB), attributes(5)

mllib_VideoColorYUV444seq_to_ABGRint(3MLIB)

NAME	mllib_VideoColorYUV444seq_to_ABGRint – color convert YUV sequential to ABGR interleaved																				
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> void mllib_VideoColorYUV444seq_to_ABGRint(mllib_u32 *ABGR, const mllib_u8 *Y, const mllib_u8 *U, const mllib_u8 *V, const mllib_u8 *A_array, mllib_u8 A_const, mllib_s32 w, mllib_s32 h, mllib_s32 dlb, mllib_s32 slb);</pre>																				
DESCRIPTION	<p>The Y, U, V pixel streams are converted into an ABGR pixel stream. All pixel components are 8-bit unsigned integers. All buffers have dimensions <i>w</i> and <i>h</i>.</p> <p>The alpha values for this function work in the following fashion:</p> <ul style="list-style-type: none">■ If <i>A_array</i> pointer is not NULL, the values are taken from there. It has to have the same dimensions as the Y buffer.■ If <i>A_array</i> pointer is NULL, the alpha values for every pixel are set to <i>A_const</i>. <p>The following equation is used:</p> $\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.1644 & 0.0000 & 1.5966 \\ 1.1644 & -0.3920 & -0.8132 \\ 1.1644 & 2.0184 & 0.0000 \end{bmatrix} * \begin{bmatrix} Y \\ U \\ V \end{bmatrix} - \begin{bmatrix} 16.0000 \\ 128.0000 \\ 128.0000 \end{bmatrix}$																				
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>ABGR</i></td><td>Pointer to output buffer.</td></tr><tr><td><i>Y</i></td><td>Pointer to Y input buffer.</td></tr><tr><td><i>U</i></td><td>Pointer to U input buffer.</td></tr><tr><td><i>V</i></td><td>Pointer to V input buffer.</td></tr><tr><td><i>A_array</i></td><td>Array of alpha values.</td></tr><tr><td><i>A_const</i></td><td>Constant alpha value.</td></tr><tr><td><i>w</i></td><td>Image width in pixels.</td></tr><tr><td><i>h</i></td><td>Image height in lines.</td></tr><tr><td><i>dlb</i></td><td>Linebytes for output buffer.</td></tr><tr><td><i>slb</i></td><td>Linebytes for input buffers.</td></tr></table>	<i>ABGR</i>	Pointer to output buffer.	<i>Y</i>	Pointer to Y input buffer.	<i>U</i>	Pointer to U input buffer.	<i>V</i>	Pointer to V input buffer.	<i>A_array</i>	Array of alpha values.	<i>A_const</i>	Constant alpha value.	<i>w</i>	Image width in pixels.	<i>h</i>	Image height in lines.	<i>dlb</i>	Linebytes for output buffer.	<i>slb</i>	Linebytes for input buffers.
<i>ABGR</i>	Pointer to output buffer.																				
<i>Y</i>	Pointer to Y input buffer.																				
<i>U</i>	Pointer to U input buffer.																				
<i>V</i>	Pointer to V input buffer.																				
<i>A_array</i>	Array of alpha values.																				
<i>A_const</i>	Constant alpha value.																				
<i>w</i>	Image width in pixels.																				
<i>h</i>	Image height in lines.																				
<i>dlb</i>	Linebytes for output buffer.																				
<i>slb</i>	Linebytes for input buffers.																				
RETURN VALUES	None.																				
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:																				

mllib_VideoColorYUV444seq_to_ABGRint(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

mllib_VideoColorYUV420seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV411seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV422seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV444seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV420seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV411seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV422seq_to_ABGRint(3MLIB),
mllib_VideoColorYUYV422int_to_ARGBint(3MLIB),
mllib_VideoColorYUV444int_to_ARGBint(3MLIB),
mllib_VideoColorYUYV422int_to_ABGRint(3MLIB),
mllib_VideoColorYUV444int_to_ABGRint(3MLIB),
mllib_VideoColorUYVY422int_to_ARGBint(3MLIB),
mllib_VideoColorUYVY422int_to_ABGRint(3MLIB),
mllib_VideoColorUYV444int_to_ARGBint(3MLIB),
mllib_VideoColorUYV444int_to_ABGRint(3MLIB), attributes(5)

mllib_VideoColorYUV444seq_to_ARGBint(3MLIB)

NAME	mllib_VideoColorYUV444seq_to_ARGBint – color convert YUV sequential to ARGB interleaved																				
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> void mllib_VideoColorYUV444seq_to_ARGBint(mllib_u32 *ARGB, const mllib_u8 *Y, const mllib_u8 *U, const mllib_u8 *V, const mllib_u8 *A_array, mllib_u8 A_const, mllib_s32 w, mllib_s32 h, mllib_s32 dlb, mllib_s32 slb);</pre>																				
DESCRIPTION	<p>The Y, U, V pixel streams are converted into an ARGB pixel stream. All pixel components are 8-bit unsigned integers. All buffers have dimensions <i>w</i> and <i>h</i>.</p> <p>The alpha values for this function work in the following fashion:</p> <ul style="list-style-type: none">■ If <i>A_array</i> pointer is not NULL, the values are taken from there. It has to have the same dimensions as the Y buffer.■ If <i>A_array</i> pointer is NULL, the alpha values for every pixel are set to <i>A_const</i>. <p>The following equation is used:</p> $\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.1644 & 0.0000 & 1.5966 \\ 1.1644 & -0.3920 & -0.8132 \\ 1.1644 & 2.0184 & 0.0000 \end{bmatrix} * \begin{bmatrix} Y \\ U \\ V \end{bmatrix} - \begin{bmatrix} 16.0000 \\ 128.0000 \\ 128.0000 \end{bmatrix}$																				
PARAMETERS	<p>The function takes the following arguments:</p> <table><tr><td><i>ARGB</i></td><td>Pointer to output buffer.</td></tr><tr><td><i>Y</i></td><td>Pointer to Y input buffer.</td></tr><tr><td><i>U</i></td><td>Pointer to U input buffer.</td></tr><tr><td><i>V</i></td><td>Pointer to V input buffer.</td></tr><tr><td><i>A_array</i></td><td>Array of alpha values.</td></tr><tr><td><i>A_const</i></td><td>Constant alpha value.</td></tr><tr><td><i>w</i></td><td>Image width in pixels.</td></tr><tr><td><i>h</i></td><td>Image height in lines.</td></tr><tr><td><i>dlb</i></td><td>Linebytes for output buffer.</td></tr><tr><td><i>slb</i></td><td>Linebytes for input buffers.</td></tr></table>	<i>ARGB</i>	Pointer to output buffer.	<i>Y</i>	Pointer to Y input buffer.	<i>U</i>	Pointer to U input buffer.	<i>V</i>	Pointer to V input buffer.	<i>A_array</i>	Array of alpha values.	<i>A_const</i>	Constant alpha value.	<i>w</i>	Image width in pixels.	<i>h</i>	Image height in lines.	<i>dlb</i>	Linebytes for output buffer.	<i>slb</i>	Linebytes for input buffers.
<i>ARGB</i>	Pointer to output buffer.																				
<i>Y</i>	Pointer to Y input buffer.																				
<i>U</i>	Pointer to U input buffer.																				
<i>V</i>	Pointer to V input buffer.																				
<i>A_array</i>	Array of alpha values.																				
<i>A_const</i>	Constant alpha value.																				
<i>w</i>	Image width in pixels.																				
<i>h</i>	Image height in lines.																				
<i>dlb</i>	Linebytes for output buffer.																				
<i>slb</i>	Linebytes for input buffers.																				
RETURN VALUES	None.																				
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:																				

mllib_VideoColorYUV444seq_to_ARGBint(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO

mllib_VideoColorYUV420seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV411seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV422seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV420seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV411seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV422seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV444seq_to_ABGRint(3MLIB),
mllib_VideoColorYUYV422int_to_ARGBint(3MLIB),
mllib_VideoColorYUV444int_to_ARGBint(3MLIB),
mllib_VideoColorYUYV422int_to_ABGRint(3MLIB),
mllib_VideoColorYUV444int_to_ABGRint(3MLIB),
mllib_VideoColorUYVY422int_to_ARGBint(3MLIB),
mllib_VideoColorUYVY422int_to_ABGRint(3MLIB),
mllib_VideoColorUYV444int_to_ARGBint(3MLIB),
mllib_VideoColorUYV444int_to_ABGRint(3MLIB), attributes(5)

mlib_VideoColorYUV444seq_to_UYVY422int(3MLIB)

NAME mlib_VideoColorYUV444seq_to_UYVY422int – convert YUV sequential to interleaved with subsampling

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

void mlib_VideoColorYUV444seq_to_UYVY422int(mlib_u32 *UYVY,
      const mlib_u8 *Y, const mlib_u8 *U, const mlib_u8 *V,
      mlib_s32 w, mlib_s32 h, mlib_s32 dlb, mlib_s32 slb);
```

DESCRIPTION The Y, U, V pixel streams are combined into a UYVY pixel stream. All pixel components are 8-bit unsigned integers. The Y, U, and V buffers have dimensions *w* and *h*. Dimension *w* is assumed to be a multiple of 2. Adjacent U and V values are averaged to get the output U and V values.

The following equation is used:

$$UYVY[r][c/2] = (((U[r][c] + U[r][c+1]) / 2) \ll 24) | (Y[r][c] \ll 16) | (((V[r][c] + V[r][c+1]) / 2) \ll 8) | (Y[r][c+1])$$

where $r = 0, 1, 2, \dots, h-1$; and $c = 0, 2, 4, \dots, w-2$.

PARAMETERS The function takes the following arguments:

UYVY Pointer to output buffer.

Y Pointer to Y input buffer.

U Pointer to U input buffer.

V Pointer to V input buffer.

w Image width in pixels.

h Image height in lines.

dlb Linebytes for output buffer.

slb Linebytes for input buffers.

RETURN VALUES None.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mlib_VideoColorYUV444seq_to_YUYV422int(3MLIB),
 mlib_VideoColorYUV444int_to_YUYV422int(3MLIB),
 mlib_VideoColorYUV444int_to_UYVY422int(3MLIB), attributes(5)

mllib_VideoColorYUV444seq_to_YUYV422int(3MLIB)

NAME | mllib_VideoColorYUV444seq_to_YUYV422int – convert YUV sequential to interleaved with subsampling

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_VideoColorYUV444seq_to_YUYV422int(mllib_u32 *YUYV,
      const mllib_u8 *Y, const mllib_u8 *U, const mllib_u8 *V,
      mllib_s32 w, mllib_s32 h, mllib_s32 dlb, mllib_s32 slb);
```

DESCRIPTION | The Y, U, V pixel streams are combined into a YUYV pixel stream. All pixel components are 8-bit unsigned integers. The Y, U, and V buffers have dimensions *w* and *h*. Dimension *w* is assumed to be a multiple of 2. Adjacent U and V values are averaged to get the output U and V values.

The following equation is used:

$$YUYV[r][c/2] = (Y[r][c] \ll 24) | ((U[r][c] + U[r][c+1]) / 2) \ll 16 | (Y[r][c+1] \ll 8) | ((V[r][c] + V[r][c+1]) / 2)$$

where $r = 0, 1, 2, \dots, h-1$; and $c = 0, 2, 4, \dots, w-2$.

PARAMETERS | The function takes the following arguments:

YUYV | Pointer to output buffer.

Y | Pointer to Y input buffer.

U | Pointer to U input buffer.

V | Pointer to V input buffer.

w | Image width in pixels.

h | Image height in lines.

dlb | Linebytes for output buffer.

slb | Linebytes for input buffers.

RETURN VALUES | None.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_VideoColorYUV444seq_to_UYVY422int(3MLIB), mllib_VideoColorYUV444int_to_YUYV422int(3MLIB), mllib_VideoColorYUV444int_to_UYVY422int(3MLIB), attributes(5)

mllib_VideoColorYUYV422int_to_ABGRint(3MLIB)

NAME mllib_VideoColorYUYV422int_to_ABGRint – color convert YUYV interleaved to ABGR interleaved

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_VideoColorYUYV422int_to_ABGRint(mllib_u32 *ABGR, const
      mllib_u32 *YUYV, const mllib_u8 *A_array, mllib_u8 A_const,
      mllib_s32 w, mllib_s32 h, mllib_s32 dlb, mllib_s32 slb, mllib_s32
      alb);
```

DESCRIPTION The YUYV pixel stream is converted into an ABGR pixel stream. All pixel components are 8-bit unsigned integers. The YUYV buffer has dimensions $w/2$ and h . The ABGR buffer has dimensions w and h . Dimensions w is assumed to be even.

The alpha values for this function work in the following fashion:

- If *A_array* pointer is not NULL, the values are taken from there. It has to have the same dimensions as the Y buffer.
- If *A_array* pointer is NULL, the alpha values for every pixel are set to *A_const*.

The following equation is used:

$$\begin{bmatrix} |R| \\ |G| \\ |B| \end{bmatrix} = \begin{bmatrix} |1.1644 & 0.0000 & 1.5966| \\ |1.1644 & -0.3920 & -0.8132| \\ |1.1644 & 2.0184 & 0.0000| \end{bmatrix} * \begin{bmatrix} [|Y| \\ |U| \\ |V| \end{bmatrix} - \begin{bmatrix} |16.0000| \\ |128.0000| \\ |128.0000| \end{bmatrix}$$

PARAMETERS The function takes the following arguments:

ABGR Pointer to output buffer.

YUYV Pointer to Y input buffer.

A_array Array of alpha values.

A_const Constant alpha value.

w Image width in pixels.

h Image height in lines.

dlb Linebytes for output buffer.

slb Linebytes for input buffer.

alb Linebytes for alpha buffer.

RETURN VALUES None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

mllib_VideoColorYUYV422int_to_ABGRint(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	MT-Safe

SEE ALSO

mllib_VideoColorYUV420seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV411seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV422seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV444seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV420seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV411seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV422seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV444seq_to_ABGRint(3MLIB),
mllib_VideoColorYUYV422int_to_ARGBint(3MLIB),
mllib_VideoColorYUV444int_to_ARGBint(3MLIB),
mllib_VideoColorYUV444int_to_ABGRint(3MLIB),
mllib_VideoColorUYVY422int_to_ARGBint(3MLIB),
mllib_VideoColorUYVY422int_to_ABGRint(3MLIB),
mllib_VideoColorUYV444int_to_ARGBint(3MLIB),
mllib_VideoColorUYV444int_to_ABGRint(3MLIB), attributes(5)

mllib_VideoColorYUYV422int_to_ARGBint(3MLIB)

NAME mllib_VideoColorYUYV422int_to_ARGBint – color convert YUYV interleaved to ARGB interleaved

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

void mllib_VideoColorYUYV422int_to_ARGBint(mllib_u32 *ARGB, const
      mllib_u32 *YUYV, const mllib_u8 *A_array, mllib_u8 A_const,
      mllib_s32 w, mllib_s32 h, mllib_s32 dlb, mllib_s32 slb, mllib_s32
      alb);
```

DESCRIPTION The YUYV pixel stream is converted into an ARGB pixel stream. All pixel components are 8-bit unsigned integers. The YUYV buffer has dimensions $w/2$ and h . The ABGR buffer has dimensions w and h . Dimensions w is assumed to be even.

The alpha values for this function work in the following fashion:

- If *A_array* pointer is not NULL, the values are taken from there. It has to have the same dimensions as the Y buffer.
- If *A_array* pointer is NULL, the alpha values for every pixel are set to *A_const*.

The following equation is used:

$$\begin{bmatrix} |R| \\ |G| \\ |B| \end{bmatrix} = \begin{bmatrix} |1.1644 & 0.0000 & 1.5966| \\ |1.1644 & -0.3920 & -0.8132| \\ |1.1644 & 2.0184 & 0.0000| \end{bmatrix} * \begin{bmatrix} [|Y| \\ |U| \\ |V| \end{bmatrix} - \begin{bmatrix} |16.0000| \\ |128.0000| \\ |128.0000| \end{bmatrix}$$

PARAMETERS The function takes the following arguments:

ARGB Pointer to output buffer.

YUYV Pointer to Y input buffer.

A_array Array of alpha values.

A_const Constant alpha value.

w Image width in pixels.

h Image height in lines.

dlb Linebytes for output buffer.

slb Linebytes for input buffer.

alb Linebytes for alpha buffer.

RETURN VALUES None.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving

mllib_VideoColorYUYV422int_to_ARGBint(3MLIB)

ATTRIBUTE TYPE	ATTRIBUTE VALUE
MT-Level	MT-Safe

SEE ALSO

mllib_VideoColorYUV420seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV411seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV422seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV444seq_to_ARGBint(3MLIB),
mllib_VideoColorYUV420seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV411seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV422seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV444seq_to_ABGRint(3MLIB),
mllib_VideoColorYUV444int_to_ARGBint(3MLIB),
mllib_VideoColorYUYV422int_to_ABGRint(3MLIB),
mllib_VideoColorYUV444int_to_ABGRint(3MLIB),
mllib_VideoColorUYVY422int_to_ARGBint(3MLIB),
mllib_VideoColorUYVY422int_to_ABGRint(3MLIB),
mllib_VideoColorUYV444int_to_ARGBint(3MLIB),
mllib_VideoColorUYV444int_to_ABGRint(3MLIB), attributes(5)

mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB)

NAME mllib_VideoCopyRefAve_U8_U8_16x16, mllib_VideoCopyRefAve_U8_U8_16x8, mllib_VideoCopyRefAve_U8_U8_8x16, mllib_VideoCopyRefAve_U8_U8_8x8, mllib_VideoCopyRefAve_U8_U8_8x4 – copies and averages a block from the reference block to the current block

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoCopyRefAve_U8_U8_16x16(mllib_u8 *curr_block,
        const mllib_u8 *ref_block, mllib_s32 stride);

mllib_status mllib_VideoCopyRefAve_U8_U8_16x8(mllib_u8 *curr_block,
        const mllib_u8 *ref_block, mllib_s32 stride);

mllib_status mllib_VideoCopyRefAve_U8_U8_8x16(mllib_u8 *curr_block,
        const mllib_u8 *ref_block, mllib_s32 stride);

mllib_status mllib_VideoCopyRefAve_U8_U8_8x8(mllib_u8 *curr_block,
        const mllib_u8 *ref_block, mllib_s32 stride);

mllib_status mllib_VideoCopyRefAve_U8_U8_8x4(mllib_u8 *curr_block,
        const mllib_u8 *ref_block, mllib_s32 stride);
```

DESCRIPTION Each of these functions copies and averages a block from the reference block to the current block. The stride applies to both the input reference block and the current block.

PARAMETERS Each of the functions takes the following arguments:

curr_block Pointer to the current block. *curr_block* must be 8-byte aligned.

ref_block Pointer to the reference block.

stride Stride, in bytes, between adjacent rows in both the current block and the reference block. *stride* must be a multiple of eight.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_VideoAddBlock_U8_S16(3MLIB)`, `mllib_VideoCopyRef_S16_U8(3MLIB)`, `mllib_VideoCopyRef_S16_U8_16x16(3MLIB)`, `mllib_VideoCopyRef_U8_U8_16x16(3MLIB)`, `mllib_VideoCopyRefAve_U8_U8(3MLIB)`, `mllib_VideoH263OverlappedMC_S16_U8(3MLIB)`, `mllib_VideoH263OverlappedMC_U8_U8(3MLIB)`,

mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB)

```
mllib_VideoInterpAveX_U8_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveXY_U8_U8(3MLIB),
mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveY_U8_U8(3MLIB),
mllib_VideoInterpAveY_U8_U8_16x16(3MLIB),
mllib_VideoInterpX_S16_U8(3MLIB),
mllib_VideoInterpX_S16_U8_16x16(3MLIB),
mllib_VideoInterpX_U8_U8(3MLIB), mllib_VideoInterpXY_S16_U8(3MLIB),
mllib_VideoInterpXY_S16_U8_16x16(3MLIB),
mllib_VideoInterpXY_U8_U8(3MLIB),
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpY_S16_U8(3MLIB),
mllib_VideoInterpY_S16_U8_16x16(3MLIB),
mllib_VideoInterpY_U8_U8(3MLIB),
mllib_VideoInterpY_U8_U8_16x16(3MLIB),
mllib_VideoP64Decimate_U8_U8(3MLIB),
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),
attributes(5)
```


NAME | mllib_VideoCopyRefAve_U8_U8 – copies and averages a block from the reference block to the current block

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoCopyRefAve_U8_U8(mllib_u8 *curr_block, const
mllib_u8 *ref_block, mllib_s32 width, mllib_s32 height, mllib_s32
stride);
```

DESCRIPTION | The mllib_VideoCopyRefAve_U8_U8() function copies and averages a block from the reference block to the current block. The stride applies to both the input reference block and the current block.

PARAMETERS | The function takes the following arguments:

curr_block | Pointer to the current block. curr_block must be 8-byte aligned.

ref_block | Pointer to the reference block.

width | Width of the blocks

height | Height of the blocks.

stride | Stride, in bytes, between adjacent rows in both the current block and the reference block. stride must be a multiple of eight.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | [mllib_VideoAddBlock_U8_S16\(3MLIB\)](#), [mllib_VideoCopyRef_S16_U8\(3MLIB\)](#), [mllib_VideoCopyRef_S16_U8_16x16\(3MLIB\)](#), [mllib_VideoCopyRef_U8_U8_16x16\(3MLIB\)](#), [mllib_VideoCopyRefAve_U8_U8_16x16\(3MLIB\)](#), [mllib_VideoH263OverlappedMC_S16_U8\(3MLIB\)](#), [mllib_VideoH263OverlappedMC_U8_U8\(3MLIB\)](#), [mllib_VideoInterpAveX_U8_U8\(3MLIB\)](#), [mllib_VideoInterpAveX_U8_U8_16x16\(3MLIB\)](#), [mllib_VideoInterpAveXY_U8_U8\(3MLIB\)](#), [mllib_VideoInterpAveXY_U8_U8_16x16\(3MLIB\)](#), [mllib_VideoInterpAveY_U8_U8\(3MLIB\)](#), [mllib_VideoInterpAveY_U8_U8_16x16\(3MLIB\)](#), [mllib_VideoInterpX_S16_U8\(3MLIB\)](#), [mllib_VideoInterpX_S16_U8_16x16\(3MLIB\)](#), [mllib_VideoInterpX_U8_U8\(3MLIB\)](#), [mllib_VideoInterpXY_S16_U8\(3MLIB\)](#),

mllib_VideoCopyRefAve_U8_U8(3MLIB)

```
mllib_VideoInterpXY_S16_U8_16x16(3MLIB),  
mllib_VideoInterpXY_U8_U8(3MLIB),  
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),  
mllib_VideoInterpY_S16_U8(3MLIB),  
mllib_VideoInterpY_S16_U8_16x16(3MLIB),  
mllib_VideoInterpY_U8_U8(3MLIB),  
mllib_VideoInterpY_U8_U8_16x16(3MLIB),  
mllib_VideoP64Decimate_U8_U8(3MLIB),  
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),  
attributes(5)
```

mlib_VideoCopyRef_S16_U8_16x16(3MLIB)

NAME mlib_VideoCopyRef_S16_U8_16x16, mlib_VideoCopyRef_S16_U8_16x8, mlib_VideoCopyRef_S16_U8_8x16, mlib_VideoCopyRef_S16_U8_8x8, mlib_VideoCopyRef_S16_U8_8x4 – copies a block from the reference block to the current block

SYNOPSIS

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_VideoCopyRef_S16_U8_16x16(mlib_s16 *mc_block,
      const mlib_u8 *ref_block, mlib_s32 stride);

mlib_status mlib_VideoCopyRef_S16_U8_16x8(mlib_s16 *mc_block,
      const mlib_u8 *ref_block, mlib_s32 stride);

mlib_status mlib_VideoCopyRef_S16_U8_8x16(mlib_s16 *mc_block,
      const mlib_u8 *ref_block, mlib_s32 stride);

mlib_status mlib_VideoCopyRef_S16_U8_8x8(mlib_s16 *mc_block, const
      mlib_u8 *ref_block, mlib_s32 stride);

mlib_status mlib_VideoCopyRef_S16_U8_8x4(mlib_s16 *mc_block, const
      mlib_u8 *ref_block, mlib_s32 stride);
```

DESCRIPTION Each of these functions copies a block from the reference block to the motion-compensated reference block. The stride applies to only the input reference block.

PARAMETERS Each of the functions takes the following arguments:

mc_block Pointer to the motion-compensated reference block. *mc_block* must be 8-byte aligned.

ref_block Pointer to the reference block.

stride Stride, in bytes, between adjacent rows in the reference block. stride must be a multiple of eight.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mlib_VideoAddBlock_U8_S16(3MLIB), mlib_VideoCopyRef_S16_U8(3MLIB), mlib_VideoCopyRef_U8_U8(3MLIB), mlib_VideoCopyRef_U8_U8_16x16(3MLIB), mlib_VideoCopyRefAve_U8_U8(3MLIB), mlib_VideoCopyRefAve_U8_U8_16x16(3MLIB),

mllib_VideoCopyRef_S16_U8_16x16(3MLIB)

```
mllib_VideoH263OverlappedMC_S16_U8(3MLIB),
mllib_VideoH263OverlappedMC_U8_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveXY_U8_U8(3MLIB),
mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveY_U8_U8(3MLIB),
mllib_VideoInterpAveY_U8_U8_16x16(3MLIB),
mllib_VideoInterpX_S16_U8(3MLIB),
mllib_VideoInterpX_S16_U8_16x16(3MLIB),
mllib_VideoInterpX_U8_U8(3MLIB), mllib_VideoInterpXY_S16_U8(3MLIB),
mllib_VideoInterpXY_S16_U8_16x16(3MLIB),
mllib_VideoInterpXY_U8_U8(3MLIB),
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpY_S16_U8(3MLIB),
mllib_VideoInterpY_S16_U8_16x16(3MLIB),
mllib_VideoInterpY_U8_U8(3MLIB),
mllib_VideoInterpY_U8_U8_16x16(3MLIB),
mllib_VideoP64Decimate_U8_U8(3MLIB),
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),
attributes(5)
```

NAME mllib_VideoCopyRef_S16_U8 – copies a block from the reference block to the current block

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoCopyRef_S16_U8(mllib_s16 *mc_block, const
    mllib_u8 *ref_block, mllib_s32 width, mllib_s32 height, mllib_s32
    stride);
```

DESCRIPTION The mllib_VideoCopyRef_S16_U8() function copies a block from the reference block to the motion-compensated reference block. The stride applies to only the input reference block.

PARAMETERS The function takes the following arguments:

mc_block Pointer to the motion-compensated reference block. *mc_block* must be 8-byte aligned.

ref_block Pointer to the reference block.

width Width of the blocks.

height Height of the blocks.

stride Stride, in bytes, between adjacent rows in the reference block. stride must be a multiple of eight.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoAddBlock_U8_S16(3MLIB),
 mllib_VideoCopyRef_S16_U8_16x16(3MLIB),
 mllib_VideoCopyRef_U8_U8(3MLIB),
 mllib_VideoCopyRef_U8_U8_16x16(3MLIB),
 mllib_VideoCopyRefAve_U8_U8(3MLIB),
 mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB),
 mllib_VideoH263OverlappedMC_S16_U8(3MLIB),
 mllib_VideoH263OverlappedMC_U8_U8(3MLIB),
 mllib_VideoInterpAveX_U8_U8(3MLIB),
 mllib_VideoInterpAveX_U8_U8_16x16(3MLIB),
 mllib_VideoInterpAveXY_U8_U8(3MLIB),
 mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB),
 mllib_VideoInterpAveY_U8_U8(3MLIB),
 mllib_VideoInterpAveY_U8_U8_16x16(3MLIB),

mllib_VideoCopyRef_S16_U8(3MLIB)

```
mllib_VideoInterpX_S16_U8(3MLIB),  
mllib_VideoInterpX_S16_U8_16x16(3MLIB),  
mllib_VideoInterpX_U8_U8(3MLIB), mllib_VideoInterpXY_S16_U8(3MLIB),  
mllib_VideoInterpXY_S16_U8_16x16(3MLIB),  
mllib_VideoInterpXY_U8_U8(3MLIB),  
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),  
mllib_VideoInterpY_S16_U8(3MLIB),  
mllib_VideoInterpY_S16_U8_16x16(3MLIB),  
mllib_VideoInterpY_U8_U8(3MLIB),  
mllib_VideoInterpY_U8_U8_16x16(3MLIB),  
mllib_VideoP64Decimate_U8_U8(3MLIB),  
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),  
attributes(5)
```

mllib_VideoCopyRef_U8_U8_16x16(3MLIB)

NAME mllib_VideoCopyRef_U8_U8_16x16, mllib_VideoCopyRef_U8_U8_16x8, mllib_VideoCopyRef_U8_U8_8x16, mllib_VideoCopyRef_U8_U8_8x8, mllib_VideoCopyRef_U8_U8_8x4 – copies an 8x8 block from the reference block to the current block

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoCopyRef_U8_U8_16x16(mllib_u8 *curr_block,
      const mllib_u8 *ref_block, mllib_s32 stride);

mllib_status mllib_VideoCopyRef_U8_U8_16x8(mllib_u8 *curr_block, const
      mllib_u8 *ref_block, mllib_s32 stride);

mllib_status mllib_VideoCopyRef_U8_U8_8x16(mllib_u8 *curr_block, const
      mllib_u8 *ref_block, mllib_s32 stride);

mllib_status mllib_VideoCopyRef_U8_U8_8x8(mllib_u8 *curr_block, const
      mllib_u8 *ref_block, mllib_s32 stride);

mllib_status mllib_VideoCopyRef_U8_U8_8x4(mllib_u8 *curr_block, const
      mllib_u8 *ref_block, mllib_s32 stride);
```

DESCRIPTION Each of these functions copies a block from the reference block to the current block. The stride applies to both the input reference block and the current block.

PARAMETERS Each of the functions takes the following arguments:

curr_block Pointer to the current block. *curr_block* must be 8-byte aligned.

ref_block Pointer to the reference block.

stride Stride, in bytes, between adjacent rows in both the current block and the reference block. *stride* must be a multiple of eight.

RETURN VALUES Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_VideoAddBlock_U8_S16(3MLIB)`, `mllib_VideoCopyRef_S16_U8(3MLIB)`, `mllib_VideoCopyRef_S16_U8_16x16(3MLIB)`, `mllib_VideoCopyRef_U8_U8(3MLIB)`, `mllib_VideoCopyRefAve_U8_U8(3MLIB)`, `mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB)`, `mllib_VideoH263OverlappedMC_S16_U8(3MLIB)`, `mllib_VideoH263OverlappedMC_U8_U8(3MLIB)`, `mllib_VideoInterpAveX_U8_U8(3MLIB)`,

mllib_VideoCopyRef_U8_U8_16x16(3MLIB)

```
mllib_VideoInterpAveX_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveXY_U8_U8(3MLIB),
mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveY_U8_U8(3MLIB),
mllib_VideoInterpAveY_U8_U8_16x16(3MLIB),
mllib_VideoInterpX_S16_U8(3MLIB),
mllib_VideoInterpX_S16_U8_16x16(3MLIB),
mllib_VideoInterpX_U8_U8(3MLIB), mllib_VideoInterpXY_S16_U8(3MLIB),
mllib_VideoInterpXY_S16_U8_16x16(3MLIB),
mllib_VideoInterpXY_U8_U8(3MLIB),
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpY_S16_U8(3MLIB),
mllib_VideoInterpY_S16_U8_16x16(3MLIB),
mllib_VideoInterpY_U8_U8(3MLIB),
mllib_VideoInterpY_U8_U8_16x16(3MLIB),
mllib_VideoP64Decimate_U8_U8(3MLIB),
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),
attributes(5)
```


NAME mllib_VideoCopyRef_U8_U8 – copies a block from the reference block to the current block

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoCopyRef_U8_U8(mllib_u8 *curr_block, const
    mllib_u8 *ref_block, mllib_s32 width, mllib_s32 height, mllib_s32
    stride);
```

DESCRIPTION The mllib_VideoCopyRef_U8_U8() function copies a block from the reference block to the current block. The stride applies to both the input reference block and the current block.

PARAMETERS The function takes the following arguments:

curr_block Pointer to the current block. *curr_block* must be 8-byte aligned.

ref_block Pointer to the reference block.

width Width of the blocks.

height Height of the blocks.

stride Stride, in bytes, between adjacent rows in both the current block and the reference block. *stride* must be a multiple of eight.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoAddBlock_U8_S16(3MLIB), mllib_VideoCopyRef_S16_U8(3MLIB), mllib_VideoCopyRef_S16_U8_16x16(3MLIB), mllib_VideoCopyRef_U8_U8_16x16(3MLIB), mllib_VideoCopyRefAve_U8_U8(3MLIB), mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB), mllib_VideoH263OverlappedMC_S16_U8(3MLIB), mllib_VideoH263OverlappedMC_U8_U8(3MLIB), mllib_VideoInterpAveX_U8_U8(3MLIB), mllib_VideoInterpAveX_U8_U8_16x16(3MLIB), mllib_VideoInterpAveXY_U8_U8(3MLIB), mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB), mllib_VideoInterpAveY_U8_U8(3MLIB), mllib_VideoInterpAveY_U8_U8_16x16(3MLIB), mllib_VideoInterpX_S16_U8(3MLIB), mllib_VideoInterpX_S16_U8_16x16(3MLIB),

mllib_VideoCopyRef_U8_U8(3MLIB)

```
mllib_VideoInterpX_U8_U8(3MLIB), mllib_VideoInterpXY_S16_U8(3MLIB),  
mllib_VideoInterpXY_S16_U8_16x16(3MLIB),  
mllib_VideoInterpXY_U8_U8(3MLIB),  
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),  
mllib_VideoInterpY_S16_U8(3MLIB),  
mllib_VideoInterpY_S16_U8_16x16(3MLIB),  
mllib_VideoInterpY_U8_U8(3MLIB),  
mllib_VideoInterpY_U8_U8_16x16(3MLIB),  
mllib_VideoP64Decimate_U8_U8(3MLIB),  
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),  
attributes(5)
```

NAME mllib_VideoDCT16x16_S16_S16 – forward Discrete Cosine Transform (DCT)

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoDCT16x16_S16_S16(mllib_s16 coeffs[256], const
mllib_s16 block[256]) ;
```

DESCRIPTION The input to the DCT routine is the difference between the current block and the reference block. The difference pixel can occupy nine bits and is represented as a 16-bit datum. The source and destination buffer addresses must be 8-byte aligned.

PARAMETERS The function takes the following arguments:

coeffs Pointer to the destination DCT coefficients. *coeffs* must be 8-byte aligned.

block Pointer to an 16x16 motion-compensated block that is the difference between the reference block and the current block. *block* must be 8-byte aligned.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoDCT2x2_S16_S16(3MLIB), mllib_VideoDCT4x4_S16_S16(3MLIB), mllib_VideoDCT8x8_S16_S16(3MLIB), mllib_VideoDCT8x8_S16_S16_B12(3MLIB), mllib_VideoDCT8x8_S16_S16_NA(3MLIB), mllib_VideoDCT8x8_S16_U8(3MLIB), mllib_VideoDCT8x8_S16_U8_NA(3MLIB), mllib_VideoDCT16x16_S16_S16(3MLIB), mllib_VideoDCT16x16_S16_S16_B10(3MLIB), mllib_VideoDeQuantize_S16(3MLIB), mllib_VideoDeQuantizeInit_S16(3MLIB), mllib_VideoQuantize_S16(3MLIB), mllib_VideoQuantizeInit_S16(3MLIB), attributes(5)

mllib_VideoDCT16x16_S16_S16_B10(3MLIB)

NAME	mllib_VideoDCT16x16_S16_S16_B10 – forward Discrete Cosine Transform (DCT)						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoDCT16x16_S16_S16_B10 (mllib_s16 coeffs[256], const mllib_s16 block[256]);</pre>						
DESCRIPTION	The mllib_VideoDCT16x16_S16_S16_B10 () function computes the forward DCT for the destination DCT coefficients of data type mllib_s16 and source block of data type mllib_s16. The input to the DCT routine is the difference between the current block and the reference block. The difference pixel which can occupy 10 bits, is represented as a 16-bit data.						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>coeffs</i> Pointer to the output DCT coefficients. Note that coeffs must be 8-byte aligned.</p> <p><i>block</i> Pointer to a 16x16 motion-compensated block which is the difference between the reference block and the current block. Note that block must be 8-byte aligned.</p>						
RETURN VALUES	The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_VideoDCT2x2_S16_S16(3MLIB), mllib_VideoDCT4x4_S16_S16(3MLIB), mllib_VideoDCT8x8_S16_S16(3MLIB), mllib_VideoDCT8x8_S16_S16_B12(3MLIB), mllib_VideoDCT8x8_S16_S16_NA(3MLIB), mllib_VideoDCT8x8_S16_U8(3MLIB), mllib_VideoDCT8x8_S16_U8_NA(3MLIB), mllib_VideoDCT16x16_S16_S16(3MLIB), mllib_VideoDeQuantize_S16(3MLIB), mllib_VideoDeQuantizeInit_S16(3MLIB), mllib_VideoQuantize_S16(3MLIB), mllib_VideoQuantizeInit_S16(3MLIB), attributes(5)						

NAME mllib_VideoDCT2x2_S16_S16 – forward Discrete Cosine Transform (DCT)

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoDCT2x2_S16_S16(mllib_s16 coeffs[4], const
mllib_s16 block[4]);
```

DESCRIPTION The mllib_VideoDCT2x2_S16_S16() function computes the forward DCT for the destination DCT coefficients of data type mllib_s16 and a source block of data type mllib_s16. The input to the DCT routine is the difference between the current block and the reference block. The difference pixel can occupy nine bits and is represented as a 16-bit datum. The source and destination buffer addresses must be 8-byte aligned.

PARAMETERS The function takes the following arguments:

coeffs Pointer to the destination DCT coefficients. coeffs must be 8-byte aligned.

block Pointer to an 2x2 motion-compensated block that is the difference between the reference block and the current block. block must be 8-byte aligned.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoDCT2x2_S16_S16(3MLIB), mllib_VideoDCT4x4_S16_S16(3MLIB), mllib_VideoDCT8x8_S16_S16(3MLIB), mllib_VideoDCT8x8_S16_S16_B12(3MLIB), mllib_VideoDCT8x8_S16_S16_NA(3MLIB), mllib_VideoDCT8x8_S16_U8(3MLIB), mllib_VideoDCT8x8_S16_U8_NA(3MLIB), mllib_VideoDCT16x16_S16_S16(3MLIB), mllib_VideoDCT16x16_S16_S16_B10(3MLIB), mllib_VideoDeQuantize_S16(3MLIB), mllib_VideoDeQuantizeInit_S16(3MLIB), mllib_VideoQuantize_S16(3MLIB), mllib_VideoQuantizeInit_S16(3MLIB), attributes(5)

mllib_VideoDCT4x4_S16_S16(3MLIB)

NAME | mllib_VideoDCT4x4_S16_S16 – forward Discrete Cosine Transform (DCT)

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoDCT4x4_S16_S16(mllib_s16 coeffs[16], const
mllib_s16 block[16]);
```

DESCRIPTION | The mllib_VideoDCT4x4_S16_S16() function computes the forward DCT for the destination DCT coefficients of data type mllib_s16 and a source block of data type mllib_s16. The input to the DCT routine is the difference between the current block and the reference block. The difference pixel can occupy nine bits and is represented as a 16-bit datum. The source and destination buffer addresses must be 8-byte aligned.

PARAMETERS | The function takes the following arguments:

coeffs | Pointer to the destination DCT coefficients. coeffs must be 8-byte aligned.

block | Pointer to an 4x4 motion-compensated block that is the difference between the reference block and the current block. block must be 8-byte aligned.

RETURN VALUES | The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_VideoDCT2x2_S16_S16(3MLIB), mllib_VideoDCT4x4_S16_S16(3MLIB), mllib_VideoDCT8x8_S16_S16(3MLIB), mllib_VideoDCT8x8_S16_S16_B12(3MLIB), mllib_VideoDCT8x8_S16_S16_NA(3MLIB), mllib_VideoDCT8x8_S16_U8(3MLIB), mllib_VideoDCT8x8_S16_U8_NA(3MLIB), mllib_VideoDCT16x16_S16_S16(3MLIB), mllib_VideoDCT16x16_S16_S16_B10(3MLIB), mllib_VideoDeQuantize_S16(3MLIB), mllib_VideoDeQuantizeInit_S16(3MLIB), mllib_VideoQuantize_S16(3MLIB), mllib_VideoQuantizeInit_S16(3MLIB), attributes(5)

NAME mllib_VideoDCT8x8_S16_S16 – forward Discrete Cosine Transform (DCT)

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoDCT8x8_S16_S16(mllib_s16 coeffs[64], const
mllib_s16 block[64]);
```

DESCRIPTION The mllib_VideoDCT8x8_S16_S16() function computes the forward DCT for the destination DCT coefficients of data type mllib_s16 and a source block of data type mllib_s16. The input to the DCT routine is the difference between the current block and the reference block. The difference pixel can occupy nine bits and is represented as a 16-bit datum. The source and destination buffer addresses must be 8-byte aligned.

PARAMETERS The function takes the following arguments:

coeffs Pointer to the destination DCT coefficients. coeffs must be 8-byte aligned.

block Pointer to an 8x8 motion-compensated block that is the difference between the reference block and the current block. block must be 8-byte aligned.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoDCT2x2_S16_S16(3MLIB), mllib_VideoDCT4x4_S16_S16(3MLIB), mllib_VideoDCT8x8_S16_S16(3MLIB), mllib_VideoDCT8x8_S16_S16_B12(3MLIB), mllib_VideoDCT8x8_S16_S16_NA(3MLIB), mllib_VideoDCT8x8_S16_U8(3MLIB), mllib_VideoDCT8x8_S16_U8_NA(3MLIB), mllib_VideoDCT16x16_S16_S16(3MLIB), mllib_VideoDCT16x16_S16_S16_B10(3MLIB), mllib_VideoDeQuantize_S16(3MLIB), mllib_VideoDeQuantizeInit_S16(3MLIB), mllib_VideoQuantize_S16(3MLIB), mllib_VideoQuantizeInit_S16(3MLIB), attributes(5)

mllib_VideoDCT8x8_S16_S16_B12(3MLIB)

NAME | mllib_VideoDCT8x8_S16_S16_B12 – forward Discrete Cosine Transform (DCT)

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]  
#include <mllib.h>  
  
mllib_status mllib_VideoDCT8x8_S16_S16_B12(mllib_s16 coeffs[64], const  
mllib_s16 block[64]) ;
```

DESCRIPTION | Purpose This function computes the forward DCT for the destination DCT coefficients of data type mllib_s16 and source block of data type mllib_s16. The source to the DCT routine is the difference between the current block and the reference block. The difference pixel, which can occupy 12 bits, is represented as a 16-bit data.

PARAMETERS | The function takes the following arguments:

coeffs | Pointer to the output DCT coefficients. Note that coeffs must be 8-byte aligned.

block | Pointer to an 8x8 motion-compensated block which is the difference between the reference block and the current block. Note that block must be 8-byte aligned.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_VideoDCT2x2_S16_S16(3MLIB), mllib_VideoDCT4x4_S16_S16(3MLIB), mllib_VideoDCT8x8_S16_S16(3MLIB), mllib_VideoDCT8x8_S16_S16_B12(3MLIB), mllib_VideoDCT8x8_S16_S16_NA(3MLIB), mllib_VideoDCT8x8_S16_U8(3MLIB), mllib_VideoDCT8x8_S16_U8_NA(3MLIB), mllib_VideoDCT16x16_S16_S16(3MLIB), mllib_VideoDCT16x16_S16_S16_B10(3MLIB), mllib_VideoDeQuantize_S16(3MLIB), mllib_VideoDeQuantizeInit_S16(3MLIB), mllib_VideoQuantize_S16(3MLIB), mllib_VideoQuantizeInit_S16(3MLIB), attributes(5)

mllib_VideoDCT8x8_S16_S16_NA(3MLIB)

NAME | mllib_VideoDCT8x8_S16_S16_NA – forward Discrete Cosine Transform (DCT)

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoDCT8x8_S16_S16_NA(mllib_s16 coeffs[64], const
mllib_s16 block[64]) ;
```

DESCRIPTION | The input to the DCT routine is the difference between the current block and the reference block. The difference pixel can occupy nine bits and is represented as a 16-bit datum. This function requires no special address alignment.

PARAMETERS | The function takes the following arguments:

coeffs | Pointer to the destination DCT coefficients.

block | Pointer to an 8x8 motion-compensated block that is the difference between the reference block and the current block.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_VideoDCT2x2_S16_S16(3MLIB), mllib_VideoDCT4x4_S16_S16(3MLIB), mllib_VideoDCT8x8_S16_S16(3MLIB), mllib_VideoDCT8x8_S16_S16_B12(3MLIB), mllib_VideoDCT8x8_S16_S16_NA(3MLIB), mllib_VideoDCT8x8_S16_U8(3MLIB), mllib_VideoDCT8x8_S16_U8_NA(3MLIB), mllib_VideoDCT16x16_S16_S16(3MLIB), mllib_VideoDCT16x16_S16_S16_B10(3MLIB), mllib_VideoDeQuantize_S16(3MLIB), mllib_VideoDeQuantizeInit_S16(3MLIB), mllib_VideoQuantize_S16(3MLIB), mllib_VideoQuantizeInit_S16(3MLIB), attributes(5)

mllib_VideoDCT8x8_S16_U8(3MLIB)

NAME | mllib_VideoDCT8x8_S16_U8 – forward Discrete Cosine Transform (DCT)

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoDCT8x8_S16_U8(mllib_s16 coeffs[64], const
mllib_u8 *block, mllib_s32 stride);
```

DESCRIPTION | The mllib_VideoDCT8x8_S16_U8() function computes the forward DCT for the destination DCT coefficients of data type mllib_s16 and source block of data type mllib_u8. The input to the DCT routine is the difference between the current block and the reference block. The difference pixel can occupy nine bits and is represented as a 16-bit datum. In the case of an mllib_s16 source data type and an mllib_u8 output data type, the stride applies to the block that is part of the frame currently being encoded.

PARAMETERS | The function takes the following arguments:

<i>coeffs</i>	Pointer to the destination DCT coefficients. coeffs must be 8-byte aligned.
<i>block</i>	Pointer to an 8x8 block in the current frame. block must be 8-byte aligned.
<i>stride</i>	Stride in bytes between adjacent rows in a block. stride must be a multiple of eight.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_VideoDCT2x2_S16_S16(3MLIB), mllib_VideoDCT4x4_S16_S16(3MLIB), mllib_VideoDCT8x8_S16_S16(3MLIB), mllib_VideoDCT8x8_S16_S16_B12(3MLIB), mllib_VideoDCT8x8_S16_S16_NA(3MLIB), mllib_VideoDCT8x8_S16_U8(3MLIB), mllib_VideoDCT8x8_S16_U8_NA(3MLIB), mllib_VideoDCT16x16_S16_S16(3MLIB), mllib_VideoDCT16x16_S16_S16_B10(3MLIB), mllib_VideoDeQuantize_S16(3MLIB), mllib_VideoDeQuantizeInit_S16(3MLIB), mllib_VideoQuantize_S16(3MLIB), mllib_VideoQuantizeInit_S16(3MLIB), attributes(5)

mllib_VideoDCT8x8_S16_U8_NA(3MLIB)

NAME | mllib_VideoDCT8x8_S16_U8_NA – forward Discrete Cosine Transform (DCT)

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoDCT8x8_S16_U8_NA(mllib_s16 coeffs[64], const
    mllib_u8 *block, mllib_s32 stride);
```

DESCRIPTION | The input to the DCT routine is the difference between the current block and the reference block. The difference pixel can occupy nine bits and is represented as a 16-bit datum. This function requires no special address alignment.

PARAMETERS | The function takes the following arguments:

coeffs | Pointer to the destination DCT coefficients.

block | Pointer to an 8x8 motion-compensated block that is the difference between the reference block and the current block.

stride | Stride in bytes between adjacent rows in a block. stride must be a multiple of eight.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_VideoDCT2x2_S16_S16(3MLIB), mllib_VideoDCT4x4_S16_S16(3MLIB), mllib_VideoDCT8x8_S16_S16(3MLIB), mllib_VideoDCT8x8_S16_S16_B12(3MLIB), mllib_VideoDCT8x8_S16_S16_NA(3MLIB), mllib_VideoDCT8x8_S16_U8(3MLIB), mllib_VideoDCT8x8_S16_U8_NA(3MLIB), mllib_VideoDCT16x16_S16_S16(3MLIB), mllib_VideoDCT16x16_S16_S16_B10(3MLIB), mllib_VideoDeQuantize_S16(3MLIB), mllib_VideoDeQuantizeInit_S16(3MLIB), mllib_VideoQuantize_S16(3MLIB), mllib_VideoQuantizeInit_S16(3MLIB), attributes(5)

mllib_VideoDeQuantizeInit_S16(3MLIB)

NAME mllib_VideoDeQuantizeInit_S16 – dequantization of forward Discrete Cosine Transform (DCT) coefficients

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoDeQuantizeInit_S16(mllib_d64 dqtable[64], const
mllib_s16 iqtable[64]);
```

DESCRIPTION The mllib_VideoDeQuantizeInit_S16() function initializes the dequantization table.

The following equation is used:

$$dqtable[i] = iqtable[i]; \quad 0 \leq i < 64$$

PARAMETERS The function takes the following arguments:

dqtable Pointer to dequantizer table coefficients.

iqtable Pointer to original quantizer table coefficients:

$$0 < iqtable[i] < 128$$

RETURN VALUES The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoDCT2x2_S16_S16(3MLIB), mllib_VideoDCT4x4_S16_S16(3MLIB), mllib_VideoDCT8x8_S16_S16(3MLIB), mllib_VideoDCT8x8_S16_S16_B12(3MLIB), mllib_VideoDCT8x8_S16_S16_NA(3MLIB), mllib_VideoDCT8x8_S16_U8(3MLIB), mllib_VideoDCT8x8_S16_U8_NA(3MLIB), mllib_VideoDCT16x16_S16_S16(3MLIB), mllib_VideoDCT16x16_S16_S16_B10(3MLIB), mllib_VideoDeQuantize_S16(3MLIB), mllib_VideoQuantize_S16(3MLIB), mllib_VideoQuantizeInit_S16(3MLIB), attributes(5)

NAME mllib_VideoDeQuantize_S16 – dequantization of forward Discrete Cosine Transform (DCT) coefficients

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoDeQuantize_S16(mllib_s16 icoeffs[64], const
    mllib_d64 dqtable[64]) ;
```

DESCRIPTION The mllib_VideoDeQuantize_S16() function performs dequantization on DCT coefficients.

The following equation is used:

$$icoeffs[i] = icoeffs[i] * dqtable[i]; \quad 0 \leq i < 64$$

PARAMETERS The function takes the following arguments:

icoeffs Pointer to the output DCT coefficients:
 -2048 < icoeffs[i] < 2048
 Note that *icoeffs* must be 8-byte aligned.

dqtable Pointer to dequantizer table coefficients.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoDCT2x2_S16_S16(3MLIB), mllib_VideoDCT4x4_S16_S16(3MLIB), mllib_VideoDCT8x8_S16_S16(3MLIB), mllib_VideoDCT8x8_S16_S16_B12(3MLIB), mllib_VideoDCT8x8_S16_S16_NA(3MLIB), mllib_VideoDCT8x8_S16_U8(3MLIB), mllib_VideoDCT8x8_S16_U8_NA(3MLIB), mllib_VideoDCT16x16_S16_S16(3MLIB), mllib_VideoDCT16x16_S16_S16_B10(3MLIB), mllib_VideoDeQuantizeInit_S16(3MLIB), mllib_VideoQuantize_S16(3MLIB), mllib_VideoQuantizeInit_S16(3MLIB), attributes(5)

mllib_VideoDownSample420(3MLIB)

NAME mllib_VideoDownSample420 – down sampling rate conversion in JFIF

SYNOPSIS cc [*flag...*] *file...* -lmllib [*library...*]
#include <mllib.h>

```
mllib_status mllib_VideoDownSample420(mllib_u8 *dst, const mllib_u8  
*src0, const mllib_u8 *src1, mllib_s32 n);
```

DESCRIPTION The mllib_VideoDownSample420() function performs down sampling rate conversion used in JPEG File Interchange Format (JFIF).

PARAMETERS The function takes the following arguments:

dst Pointer to destination row. dst must be 8-byte aligned.

src0 Pointer to upper source row. src0 must be 8-byte aligned.

src1 Pointer to middle source row. src1 must be 8-byte aligned.

n Length of source rows. n must be greater than 1.

RETURN VALUES The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoDownSample420_S16(3MLIB), mllib_VideoDownSample422(3MLIB), mllib_VideoDownSample422_S16(3MLIB), mllib_VideoUpSample420(3MLIB), mllib_VideoUpSample420_Nearest(3MLIB), mllib_VideoUpSample420_Nearest_S16(3MLIB), mllib_VideoUpSample420_S16(3MLIB), mllib_VideoUpSample422(3MLIB), mllib_VideoUpSample422_S16(3MLIB), mllib_VideoUpSample422_Nearest(3MLIB), mllib_VideoUpSample422_Nearest_S16(3MLIB), attributes(5)

mlib_VideoDownSample420_S16(3MLIB)

NAME | mlib_VideoDownSample420_S16 – down sampling rate conversion in JFIF

SYNOPSIS |

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_VideoDownSample420_S16(mlib_s16 *dst, const
mlib_s16 *src0, const mlib_s16 *src1, mlib_s32 n);
```

DESCRIPTION | The mlib_VideoDownSample420_S16() function performs down sampling rate conversion used in JPEG File Interchange Format (JFIF).

PARAMETERS | The function takes the following arguments:

dst | Pointer to destination row. *dst* must be 8-byte aligned.

src0 | Pointer to upper source row. *src0* must be 8-byte aligned.

src1 | Pointer to middle source row. *src1* must be 8-byte aligned.

n | Length of source rows. *n* must be greater than 1.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mlib_VideoDownSample420(3MLIB), mlib_VideoDownSample422(3MLIB), mlib_VideoDownSample422_S16(3MLIB), mlib_VideoUpSample420(3MLIB), mlib_VideoUpSample420_Nearest(3MLIB), mlib_VideoUpSample420_Nearest_S16(3MLIB), mlib_VideoUpSample420_S16(3MLIB), mlib_VideoUpSample422(3MLIB), mlib_VideoUpSample422_S16(3MLIB), mlib_VideoUpSample422_Nearest(3MLIB), mlib_VideoUpSample422_Nearest_S16(3MLIB), attributes(5)

mllib_VideoDownSample422(3MLIB)

NAME mllib_VideoDownSample422 – down sampling rate conversion in JFIF

SYNOPSIS
cc [*flag...*] *file...* -lmllib [*library...*]
#include <mllib.h>

```
mllib_status mllib_VideoDownSample422(mllib_u8 *dst, const mllib_u8
    *src, mllib_s32 n);
```

DESCRIPTION The mllib_VideoDownSample422() function performs down sampling rate conversion used in JPEG File Interchange Format (JFIF).

PARAMETERS The function takes the following arguments:

dst Pointer to destination row. dst must be 8-byte aligned.

src Pointer to source row. src must be 8-byte aligned.

n Length of source rows. n must be greater than 1.

RETURN VALUES The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoDownSample420(3MLIB), mllib_VideoDownSample420_S16(3MLIB), mllib_VideoDownSample422_S16(3MLIB), mllib_VideoUpSample420(3MLIB), mllib_VideoUpSample420_Nearest(3MLIB), mllib_VideoUpSample420_Nearest_S16(3MLIB), mllib_VideoUpSample420_S16(3MLIB), mllib_VideoUpSample422(3MLIB), mllib_VideoUpSample422_S16(3MLIB), mllib_VideoUpSample422_Nearest(3MLIB), mllib_VideoUpSample422_Nearest_S16(3MLIB), attributes(5)

mllib_VideoDownSample422_S16(3MLIB)

NAME | mllib_VideoDownSample422_S16 – down sampling rate conversion in JFIF

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoDownSample422_S16(mllib_s16 *dst, const
mllib_s16 *src, mllib_s32 n);
```

DESCRIPTION | The `mllib_VideoDownSample422_S16()` function performs down sampling rate conversion used in JPEG File Interchange Format (JFIF).

PARAMETERS | The function takes the following arguments:

dst | Pointer to destination row. *dst* must be 8-byte aligned.

src | Pointer to source row. *src* must be 8-byte aligned.

n | Length of source rows. *n* must be greater than 1.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_VideoDownSample420(3MLIB)`, `mllib_VideoDownSample420_S16(3MLIB)`, `mllib_VideoDownSample422(3MLIB)`, `mllib_VideoUpSample420(3MLIB)`, `mllib_VideoUpSample420_Nearest(3MLIB)`, `mllib_VideoUpSample420_Nearest_S16(3MLIB)`, `mllib_VideoUpSample420_S16(3MLIB)`, `mllib_VideoUpSample422(3MLIB)`, `mllib_VideoUpSample422_S16(3MLIB)`, `mllib_VideoUpSample422_Nearest(3MLIB)`, `mllib_VideoUpSample422_Nearest_S16(3MLIB)`, `attributes(5)`

mllib_VideoH263OverlappedMC_S16_U8(3MLIB)

NAME	mllib_VideoH263OverlappedMC_S16_U8 – generates the 8x8 luminance prediction block in the Advanced Prediction Mode for H.263 codec
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoH263OverlappedMC_S16_U8(mllib_s16 mc_block[64], const mllib_u8 *ref_frame, mllib_s32 mch, mllib_s32 mcv, mllib_s32 mah, mllib_s32 mav, mllib_s32 mbh, mllib_s32 mbv, mllib_s32 mlh, mllib_s32 mlv, mllib_s32 mrh, mllib_s32 mrv, mllib_s32 ref_stride);</pre>
DESCRIPTION	<p>The mllib_VideoH263OverlappedMC_S16_U8() function generates an 8x8 luminance prediction block (motion-compensated block) in the Advanced Prediction Mode for H.263 codec. The reference frame in this function is an interpolated frame. The output of this function must be added to the IDCT output in order to reconstruct the block in the current frame.</p> <p>The following equation is used:</p> <p>for $x = 0, 1, 2, 3; y = 0, 1, 2, 3$</p> $mc(x, y) = (ref(2x + mch, 2y + mcv)*H0(x, y) + ref(2x + mah, 2y + mav)*H1(x, y) + ref(2x + mlh, 2y + mlv)*H2(x, y)) / 8;$ <p>for $x = 4, 5, 6, 7; y = 0, 1, 2, 3$</p> $mc(x, y) = (ref(2x + mch, 2y + mcv)*H0(x, y) + ref(2x + mah, 2y + mav)*H1(x, y) + ref(2x + mrh, 2y + mrv)*H2(x, y)) / 8;$ <p>for $x = 0, 1, 2, 3; y = 4, 5, 6, 7$</p> $mc(x, y) = (ref(2x + mch, 2y + mcv)*H0(x, y) + ref(2x + mbh, 2y + mbv)*H1(x, y) + ref(2x + mlh, 2y + mlv)*H2(x, y)) / 8;$ <p>for $x = 4, 5, 6, 7; y = 4, 5, 6, 7$</p> $mc(x, y) = (ref(2x + mch, 2y + mcv)*H0(x, y) + ref(2x + mbh, 2y + mbv)*H1(x, y) + ref(2x + mrh, 2y + mrv)*H2(x, y)) / 8;$ <p>where</p> $H0 = \begin{bmatrix} 4 & 5 & 5 & 5 & 5 & 5 & 5 & 4 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 6 & 6 & 6 & 6 & 5 & 5 \\ 5 & 5 & 6 & 6 & 6 & 6 & 5 & 5 \\ 5 & 5 & 6 & 6 & 6 & 6 & 5 & 5 \\ 5 & 5 & 6 & 6 & 6 & 6 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 \\ 4 & 5 & 5 & 5 & 5 & 5 & 5 & 4 \end{bmatrix}$ $H1 = \begin{bmatrix} 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 1 & 1 & 2 & 2 & 2 & 2 & 1 & 1 \end{bmatrix}$

```

H1 = | 1 1 1 1 1 1 1 1 |
      | 1 1 1 1 1 1 1 1 |
      | 1 1 1 1 1 1 1 1 |
      | 1 1 1 1 1 1 1 1 |
      | 1 1 2 2 2 2 1 1 |
      [ 2 2 2 2 2 2 2 2 ]

      [ 2 1 1 1 1 1 1 2 ]
      | 2 2 1 1 1 1 2 2 |
      | 2 2 1 1 1 1 2 2 |
H2 = | 2 2 1 1 1 1 2 2 |
      | 2 2 1 1 1 1 2 2 |
      | 2 2 1 1 1 1 2 2 |
      | 2 2 1 1 1 1 2 2 |
      | 2 2 1 1 1 1 2 2 |
      [ 2 1 1 1 1 1 1 2 ]

```

PARAMETERS The function takes the following arguments:

<i>mc_block</i>	Pointer to the motion-compensated block.
<i>ref_frame</i>	Pointer to the interpolated reference frame.
<i>mch</i>	Horizontal coordinate of the motion vector for the current block.
<i>mcv</i>	Vertical coordinate of the motion vector for the current block.
<i>mah</i>	Horizontal coordinate of the motion vector for the block above the current block.
<i>mav</i>	Vertical coordinate of the motion vector for the block above the current block.
<i>mbh</i>	Horizontal coordinate of the motion vector for the block below the current block.
<i>mbv</i>	Vertical coordinate of the motion vector for the block below the current block.
<i>mlh</i>	Horizontal coordinate of the motion vector for the block to the left of the current block.
<i>mlv</i>	Vertical coordinate of the motion vector for the block to the left of the current block.
<i>mrh</i>	Horizontal coordinate of the motion vector for the block to the right of the current block.
<i>mrv</i>	Vertical coordinate of the motion vector for the block to the right of the current block.
<i>ref_stride</i>	Stride, in bytes, between adjacent rows in the interpolated reference frame.

RETURN VALUES The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

mllib_VideoH263OverlappedMC_S16_U8(3MLIB)

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoAddBlock_U8_S16(3MLIB), mllib_VideoCopyRef_S16_U8(3MLIB),
mllib_VideoCopyRef_S16_U8_16x16(3MLIB),
mllib_VideoCopyRef_U8_U8_16x16(3MLIB),
mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB),
mllib_VideoH263OverlappedMC_U8_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveXY_U8_U8(3MLIB),
mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveY_U8_U8(3MLIB),
mllib_VideoInterpAveY_U8_U8_16x16(3MLIB),
mllib_VideoInterpX_S16_U8(3MLIB),
mllib_VideoInterpX_S16_U8_16x16(3MLIB),
mllib_VideoInterpX_U8_U8(3MLIB), mllib_VideoInterpXY_S16_U8(3MLIB),
mllib_VideoInterpXY_S16_U8_16x16(3MLIB),
mllib_VideoInterpXY_U8_U8(3MLIB),
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpY_S16_U8(3MLIB),
mllib_VideoInterpY_S16_U8_16x16(3MLIB),
mllib_VideoInterpY_U8_U8(3MLIB),
mllib_VideoInterpY_U8_U8_16x16(3MLIB),
mllib_VideoP64Decimate_U8_U8(3MLIB),
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),
attributes(5)

mlib_VideoH263OverlappedMC_U8_U8(3MLIB)

NAME	mlib_VideoH263OverlappedMC_U8_U8 – generates the 8x8 luminance prediction block in the Advanced Prediction Mode for H.263 codec
SYNOPSIS	<pre>cc [flag...] file... -lmlib [library...] #include <mlib.h> mlib_status mlib_VideoH263OverlappedMC_U8_U8(mlib_u8 *curr_block, const mlib_u8 *ref_frame, mlib_s32 mch, mlib_s32 mcv, mlib_s32 mah, mlib_s32 mav, mlib_s32 mbh, mlib_s32 mbv, mlib_s32 mlh, mlib_s32 mlv, mlib_s32 mrh, mlib_s32 mrv, mlib_s32 curr_stride, mlib_s32 ref_stride);</pre>
DESCRIPTION	<p>The mlib_VideoH263OverlappedMC_U8_U8() function generates an 8x8 luminance prediction block (motion-compensated block) in the Advanced Prediction Mode for H.263 codec. The reference frame in this function is an interpolated frame.</p> <p>The following equation is used:</p> <pre>for x = 0, 1, 2, 3; y = 0, 1, 2, 3 curr(x, y) = (ref(2x + mch, 2y + mcv)*H0(x, y) + ref(2x + mah, 2y + mav)*H1(x, y) + ref(2x + mlh, 2y + mlv)*H2(x, y)) / 8;</pre> <pre>for x = 4, 5, 6, 7; y = 0, 1, 2, 3 curr(x, y) = (ref(2x + mch, 2y + mcv)*H0(x, y) + ref(2x + mah, 2y + mav)*H1(x, y) + ref(2x + mrh, 2y + mrv)*H2(x, y)) / 8;</pre> <pre>for x = 0, 1, 2, 3; y = 4, 5, 6, 7 curr(x, y) = (ref(2x + mch, 2y + mcv)*H0(x, y) + ref(2x + mbh, 2y + mbv)*H1(x, y) + ref(2x + mlh, 2y + mlv)*H2(x, y)) / 8;</pre> <pre>for x = 4, 5, 6, 7; y = 4, 5, 6, 7 curr(x, y) = (ref(2x + mch, 2y + mcv)*H0(x, y) + ref(2x + mbh, 2y + mbv)*H1(x, y) + ref(2x + mrh, 2y + mrv)*H2(x, y)) / 8;</pre> <p>where</p> <pre> [4 5 5 5 5 5 5 4] 5 5 5 5 5 5 5 5 5 5 6 6 6 6 5 5 H0 = 5 5 6 6 6 6 5 5 5 5 6 6 6 6 5 5 5 5 6 6 6 6 5 5 5 5 5 5 5 5 5 5 [4 5 5 5 5 5 5 4] [2 2 2 2 2 2 2 2] 1 1 2 2 2 2 1 1 1 1 1 1 1 1 1 1 H1 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 </pre>

mllib_VideoH263OverlappedMC_U8_U8(3MLIB)

$$\begin{array}{c}
 \begin{array}{|cccccccc|}
 \hline
 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
 \hline
 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
 \hline
 1 & 1 & 2 & 2 & 2 & 2 & 1 & 1 \\
 \hline
 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\
 \hline
 \end{array} \\
 \\
 \begin{array}{|cccccccc|}
 \hline
 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 \\
 \hline
 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\
 \hline
 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\
 \hline
 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\
 \hline
 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\
 \hline
 2 & 2 & 1 & 1 & 1 & 1 & 2 & 2 \\
 \hline
 2 & 1 & 1 & 1 & 1 & 1 & 1 & 2 \\
 \hline
 \end{array} \\
 \text{H2} =
 \end{array}$$

PARAMETERS

The function takes the following arguments:

<i>curr_block</i>	Pointer to the current block.
<i>ref_frame</i>	Pointer to the interpolated reference frame.
<i>mch</i>	Horizontal coordinate of the motion vector for the current block.
<i>mcv</i>	Vertical coordinate of the motion vector for the current block.
<i>mah</i>	Horizontal coordinate of the motion vector for the block above the current block.
<i>mav</i>	Vertical coordinate of the motion vector for the block above the current block.
<i>mbh</i>	Horizontal coordinate of the motion vector for the block below the current block.
<i>mbv</i>	Vertical coordinate of the motion vector for the block below the current block.
<i>mlh</i>	Horizontal coordinate of the motion vector for the block to the left of the current block.
<i>mlv</i>	Vertical coordinate of the motion vector for the block to the left of the current block.
<i>mrh</i>	Horizontal coordinate of the motion vector for the block to the right of the current block.
<i>mrv</i>	Vertical coordinate of the motion vector for the block to the right of the current block.
<i>curr_stride</i>	Stride, in bytes, between adjacent rows in the current frame.
<i>ref_stride</i>	Stride, in bytes, between adjacent rows in the interpolated reference frame.

RETURN VALUES

The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

mllib_VideoH263OverlappedMC_U8_U8(3MLIB)

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoAddBlock_U8_S16(3MLIB), mllib_VideoCopyRef_S16_U8(3MLIB),
mllib_VideoCopyRef_S16_U8_16x16(3MLIB),
mllib_VideoCopyRef_U8_U8_16x16(3MLIB),
mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB),
mllib_VideoH263OverlappedMC_S16_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveXY_U8_U8(3MLIB),
mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveY_U8_U8(3MLIB),
mllib_VideoInterpAveY_U8_U8_16x16(3MLIB),
mllib_VideoInterpX_S16_U8(3MLIB),
mllib_VideoInterpX_S16_U8_16x16(3MLIB),
mllib_VideoInterpX_U8_U8(3MLIB), mllib_VideoInterpXY_S16_U8(3MLIB),
mllib_VideoInterpXY_S16_U8_16x16(3MLIB),
mllib_VideoInterpXY_U8_U8(3MLIB),
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpY_S16_U8(3MLIB),
mllib_VideoInterpY_S16_U8_16x16(3MLIB),
mllib_VideoInterpY_U8_U8(3MLIB),
mllib_VideoInterpY_U8_U8_16x16(3MLIB),
mllib_VideoP64Decimate_U8_U8(3MLIB),
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),
attributes(5)

mllib_VideoIDCT8x8_S16_S16(3MLIB)

NAME	mllib_VideoIDCT8x8_S16_S16 – inverse Discrete Cosine Transform						
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_VideoIDCT8x8_S16_S16(mllib_s16 <i>block</i>[64], const mllib_s16 <i>coeffs</i>[64]);</pre>						
DESCRIPTION	The mllib_VideoIDCT8x8_S16_S16() function computes the inverse DCT (called IDCT) for the output IDCT block of data type mllib_s16 and input DCT coefficients of data type mllib_s16. This function is not guaranteed to be IEEE-1180-compliant. The output of the IDCT routine is the difference between the current block and the reference block. The difference pixel can occupy nine bits and is represented as a 16-bit datum. The output must be added to the motion-compensated reference block in order to reconstruct the current block.						
PARAMETERS	The function takes the following arguments: <i>block</i> Pointer to an 8x8 motion-compensated block that is the difference between the reference block and the current block. <i>block</i> must be 8-byte aligned. <i>coeffs</i> Pointer to the source DCT coefficients. <i>coeffs</i> must be 8-byte aligned.						
RETURN VALUES	The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_VideoIDCT_IEEE_S16_S16(3MLIB), mllib_VideoIDCT8x8_S16_S16_DC(3MLIB), mllib_VideoIDCT8x8_S16_S16_NA(3MLIB), mllib_VideoIDCT8x8_S16_S16_Q1(3MLIB), mllib_VideoIDCT8x8_U8_S16(3MLIB), mllib_VideoIDCT8x8_U8_S16_DC(3MLIB), mllib_VideoIDCT8x8_U8_S16_NA(3MLIB), mllib_VideoIDCT8x8_U8_S16_Q1(3MLIB), attributes(5)						

NAME mllib_VideoIDCT8x8_S16_S16_DC – inverse Discrete Cosine Transform

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoIDCT8x8_S16_S16_DC(mllib_s16 *block, const
mllib_s16 *coeffs);
```

DESCRIPTION The mllib_VideoIDCT8x8_S16_S16_DC() function can be used only when $F(0,0)$ is nonzero. It computes the inverse DCT (called IDCT) for the output IDCT block of data type mllib_s16 and input DCT coefficients of data type mllib_s16. This function is not guaranteed to be IEEE-1180-compliant. The output of the IDCT routine is the difference between the current block and the reference block. The difference pixel can occupy nine bits and is represented as a 16-bit datum. The output must be added to the motion-compensated reference block in order to reconstruct the current block.

PARAMETERS The function takes the following arguments:

block Pointer to the current block. *block* must be 8-byte aligned.

coeffs Pointer to the source DCT coefficients. *coeffs* must be 8-byte aligned.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoIDCT_IEEE_S16_S16(3MLIB),
mllib_VideoIDCT8x8_S16_S16(3MLIB),
mllib_VideoIDCT8x8_S16_S16_NA(3MLIB),
mllib_VideoIDCT8x8_S16_S16_Q1(3MLIB),
mllib_VideoIDCT8x8_U8_S16(3MLIB),
mllib_VideoIDCT8x8_U8_S16_DC(3MLIB),
mllib_VideoIDCT8x8_U8_S16_NA(3MLIB),
mllib_VideoIDCT8x8_U8_S16_Q1(3MLIB), attributes(5)

mllib_VideoIDCT8x8_S16_S16_NA(3MLIB)

NAME	mllib_VideoIDCT8x8_S16_S16_NA – inverse Discrete Cosine Transform						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoIDCT8x8_S16_S16_NA(mllib_s16 block[64], const mllib_s16 coeffs[64]);</pre>						
DESCRIPTION	The <code>mllib_VideoIDCT8x8_S16_S16_NA()</code> function computes the inverse DCT (called IDCT) for the output IDCT block of data type <code>mllib_s16</code> and input DCT coefficients of data type <code>mllib_s16</code> . This function is not guaranteed to be IEEE-1180-compliant. The output of the IDCT routine is the difference between the current block and the reference block. The difference pixel can occupy nine bits and is represented as a 16-bit datum. The output must be added to the motion-compensated reference block in order to reconstruct the current block.						
PARAMETERS	The function takes the following arguments: <table><tr><td><i>block</i></td><td>Pointer to the current block.</td></tr><tr><td><i>coeffs</i></td><td>Pointer to the source DCT coefficients. <i>coeffs</i> need not be 8-byte aligned.</td></tr></table>	<i>block</i>	Pointer to the current block.	<i>coeffs</i>	Pointer to the source DCT coefficients. <i>coeffs</i> need not be 8-byte aligned.		
<i>block</i>	Pointer to the current block.						
<i>coeffs</i>	Pointer to the source DCT coefficients. <i>coeffs</i> need not be 8-byte aligned.						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_VideoIDCT_IEEE_S16_S16(3MLIB)</code> , <code>mllib_VideoIDCT8x8_S16_S16(3MLIB)</code> , <code>mllib_VideoIDCT8x8_S16_S16_DC(3MLIB)</code> , <code>mllib_VideoIDCT8x8_S16_S16_Q1(3MLIB)</code> , <code>mllib_VideoIDCT8x8_U8_S16(3MLIB)</code> , <code>mllib_VideoIDCT8x8_U8_S16_DC(3MLIB)</code> , <code>mllib_VideoIDCT8x8_U8_S16_NA(3MLIB)</code> , <code>mllib_VideoIDCT8x8_U8_S16_Q1(3MLIB)</code> , <code>attributes(5)</code>						

mllib_VideoIDCT8x8_S16_S16_Q1(3MLIB)

NAME mllib_VideoIDCT8x8_S16_S16_Q1 – inverse Discrete Cosine Transform

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoIDCT8x8_S16_S16_Q1(mllib_s16 block[64], const
mllib_s16 coeffs[64]);
```

DESCRIPTION The mllib_VideoIDCT8x8_S16_S16_Q1() function can be used only when F(u,v) are nonzero for 0 ≤ u < 4 and 0 ≤ v < 4.

PARAMETERS The function takes the following arguments:

block Pointer to the current block. block must be 8-byte aligned.

coeffs Pointer to the source DCT coefficients. coeffs must be 8-byte aligned.

RETURN VALUES The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoIDCT_IEEE_S16_S16(3MLIB),
mllib_VideoIDCT8x8_S16_S16(3MLIB),
mllib_VideoIDCT8x8_S16_S16_DC(3MLIB),
mllib_VideoIDCT8x8_S16_S16_NA(3MLIB),
mllib_VideoIDCT8x8_U8_S16(3MLIB),
mllib_VideoIDCT8x8_U8_S16_DC(3MLIB),
mllib_VideoIDCT8x8_U8_S16_NA(3MLIB),
mllib_VideoIDCT8x8_U8_S16_Q1(3MLIB), attributes(5)

mllib_VideoIDCT8x8_S16_S16_Q1_Mismatch(3MLIB)

NAME	mllib_VideoIDCT8x8_S16_S16_Q1_Mismatch – inverse Discrete Cosine Transform						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoIDCT8x8_S16_S16_Q1_Mismatch(mllib_s16 block[64], const mllib_s16 coeffs[64]);</pre>						
DESCRIPTION	<p>The <code>mllib_VideoIDCT8x8_S16_S16_Q1_Mismatch()</code> function computes the inverse IDCT in the inter mode.</p> <p>This function is similar to <code>mllib_VideoIDCT8x8_S16_S16_Q1()</code> which should only be used when <code>coeffs[u][v]</code> ($u, v = 0 \dots 7$) are non-zero only for u and v less than 4. However, this function also allows element <code>coeffs[7][7]</code> to be non-zero. The primary benefit of this modification is that it can handle situations where <code>coeffs[7][7]</code> has been made non-zero by MPEG mismatch-control, allowing a simplified version of the IDCT to be undertaken for a much larger number of situations.</p>						
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>block</i> Pointer to an 8x8 motion-compensated block which is the difference between the reference block and current block. <i>block</i> must be 8-byte aligned.</p> <p><i>coeffs</i> Pointer to the input DCT coefficients. <i>coeffs</i> must be 8-byte aligned. <i>coeffs</i> should be in S12 range or it should be obtained from the correspondent direct DCT.</p>						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:						
	<table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_VideoIDCT8x8_S16_S16_Q1(3MLIB)</code> , <code>attributes(5)</code>						

NAME mllib_VideoIDCT8x8_U8_S16 – inverse Discrete Cosine Transform

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoIDCT8x8_U8_S16(mllib_u8 *block, const
    mllib_s16 coeffs[64], mllib_s32 stride);
```

DESCRIPTION The mllib_VideoIDCT8x8_U8_S16() function computes the inverse DCT (called IDCT) for the destination IDCT block of data type mllib_u8 and source DCT coefficients of data type mllib_s16.

The stride applies to the block that is part of the frame currently being reconstructed.

PARAMETERS The function takes the following arguments:

block Pointer to an 8x8 block in the current frame. block must be 8-byte aligned.

coeffs Pointer to the source DCT coefficients. coeffs must be 8-byte aligned.

stride Stride, in bytes, between adjacent rows in a block. stride must be a multiple of eight.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoIDCT_IEEE_S16_S16(3MLIB), mllib_VideoIDCT8x8_S16_S16(3MLIB), mllib_VideoIDCT8x8_S16_S16_DC(3MLIB), mllib_VideoIDCT8x8_S16_S16_NA(3MLIB), mllib_VideoIDCT8x8_S16_S16_Q1(3MLIB), mllib_VideoIDCT8x8_U8_S16_DC(3MLIB), mllib_VideoIDCT8x8_U8_S16_NA(3MLIB), mllib_VideoIDCT8x8_U8_S16_Q1(3MLIB), attributes(5)

mllib_VideoIDCT8x8_U8_S16_DC(3MLIB)

NAME	mllib_VideoIDCT8x8_U8_S16_DC – inverse Discrete Cosine Transform						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoIDCT8x8_U8_S16_DC(mllib_u8 *block, const mllib_s16 *coeffs, mllib_s32 stride);</pre>						
DESCRIPTION	The <code>mllib_VideoIDCT8x8_U8_S16_DC()</code> function can be used only when $F(0,0)$ is nonzero. It computes the inverse DCT (called IDCT) for the destination IDCT block of data type <code>mllib_u8</code> and source DCT coefficients of data type <code>mllib_s16</code> . This function is not guaranteed to be IEEE-1180-compliant. The output of the IDCT routine is the difference between the current block and the reference block. The difference pixel can occupy nine bits and is represented as a 16-bit datum. The output must be added to the motion-compensated reference block in order to reconstruct the current block.						
PARAMETERS	The function takes the following arguments: <table><tr><td><i>block</i></td><td>Pointer to the current block. <i>block</i> must be 8-byte aligned.</td></tr><tr><td><i>coeffs</i></td><td>Pointer to the source DCT coefficients. <i>coeffs</i> must be 8-byte aligned.</td></tr><tr><td><i>stride</i></td><td>Stride, in bytes, between adjacent rows in a block. <i>stride</i> must be a multiple of eight..</td></tr></table>	<i>block</i>	Pointer to the current block. <i>block</i> must be 8-byte aligned.	<i>coeffs</i>	Pointer to the source DCT coefficients. <i>coeffs</i> must be 8-byte aligned.	<i>stride</i>	Stride, in bytes, between adjacent rows in a block. <i>stride</i> must be a multiple of eight..
<i>block</i>	Pointer to the current block. <i>block</i> must be 8-byte aligned.						
<i>coeffs</i>	Pointer to the source DCT coefficients. <i>coeffs</i> must be 8-byte aligned.						
<i>stride</i>	Stride, in bytes, between adjacent rows in a block. <i>stride</i> must be a multiple of eight..						
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .						
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	<code>mllib_VideoIDCT_IEEE_S16_S16(3MLIB)</code> , <code>mllib_VideoIDCT8x8_S16_S16(3MLIB)</code> , <code>mllib_VideoIDCT8x8_S16_S16_DC(3MLIB)</code> , <code>mllib_VideoIDCT8x8_S16_S16_NA(3MLIB)</code> , <code>mllib_VideoIDCT8x8_S16_S16_Q1(3MLIB)</code> , <code>mllib_VideoIDCT8x8_U8_S16(3MLIB)</code> , <code>mllib_VideoIDCT8x8_U8_S16_NA(3MLIB)</code> , <code>mllib_VideoIDCT8x8_U8_S16_Q1(3MLIB)</code> , <code>attributes(5)</code>						

mllib_VideoIDCT8x8_U8_S16_NA(3MLIB)

NAME | mllib_VideoIDCT8x8_U8_S16_NA – inverse Discrete Cosine Transform

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoIDCT8x8_U8_S16_NA(mllib_u8 *block, const
      mllib_s16 coeffs[64], mllib_s32 stride);
```

DESCRIPTION | The mllib_VideoIDCT8x8_U8_S16_NA() function computes the inverse DCT (called IDCT) for the destination IDCT block of data type mllib_u8 and source DCT coefficients of data type mllib_s16.

The stride applies to the block that is part of the frame currently being reconstructed.

PARAMETERS | The function takes the following arguments:

block | Pointer to the current block.

coeffs | Pointer to the source DCT coefficients. coeffs need not be 8-byte aligned.

stride | Stride, in bytes, between adjacent rows in a block. stride must be a multiple of eight.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_VideoIDCT_IEEE_S16_S16(3MLIB),
mllib_VideoIDCT8x8_S16_S16(3MLIB),
mllib_VideoIDCT8x8_S16_S16_DC(3MLIB),
mllib_VideoIDCT8x8_S16_S16_NA(3MLIB),
mllib_VideoIDCT8x8_S16_S16_Q1(3MLIB),
mllib_VideoIDCT8x8_U8_S16(3MLIB),
mllib_VideoIDCT8x8_U8_S16_DC(3MLIB),
mllib_VideoIDCT8x8_U8_S16_Q1(3MLIB), attributes(5)

mllib_VideoIDCT8x8_U8_S16_Q1(3MLIB)

NAME	mllib_VideoIDCT8x8_U8_S16_Q1 – inverse Discrete Cosine Transform						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoIDCT8x8_U8_S16_Q1(mllib_u8 *block, const mllib_s16 coeffs[64], mllib_s32 stride);</pre>						
DESCRIPTION	The mllib_VideoIDCT8x8_U8_S16_Q1() function can be used only when $F(u, v)$ are nonzero and only when $0 \leq u < 4$ and $0 \leq v < 4$. The stride applies to the block that is part of the frame currently being reconstructed.						
PARAMETERS	The function takes the following arguments: <table><tr><td><i>block</i></td><td>Pointer to the current block. block must be 8-byte aligned</td></tr><tr><td><i>coeffs</i></td><td>Pointer to the source DCT coefficients. coeffs must be 8-byte aligned.</td></tr><tr><td><i>stride</i></td><td>Stride, in bytes, between adjacent rows in a block. stride must be a multiple of eight.</td></tr></table>	<i>block</i>	Pointer to the current block. block must be 8-byte aligned	<i>coeffs</i>	Pointer to the source DCT coefficients. coeffs must be 8-byte aligned.	<i>stride</i>	Stride, in bytes, between adjacent rows in a block. stride must be a multiple of eight.
<i>block</i>	Pointer to the current block. block must be 8-byte aligned						
<i>coeffs</i>	Pointer to the source DCT coefficients. coeffs must be 8-byte aligned.						
<i>stride</i>	Stride, in bytes, between adjacent rows in a block. stride must be a multiple of eight.						
RETURN VALUES	The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_VideoIDCT_IEEE_S16_S16(3MLIB), mllib_VideoIDCT8x8_S16_S16(3MLIB), mllib_VideoIDCT8x8_S16_S16_DC(3MLIB), mllib_VideoIDCT8x8_S16_S16_NA(3MLIB), mllib_VideoIDCT8x8_S16_S16_Q1(3MLIB), mllib_VideoIDCT8x8_U8_S16(3MLIB), mllib_VideoIDCT8x8_U8_S16_DC(3MLIB), mllib_VideoIDCT8x8_U8_S16_NA(3MLIB), attributes(5)						

NAME mllib_VideoIDCT_IEEE_S16_S16 – IEEE-1180 compliant inverse Discrete Cosine Transform

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoIDCT_IEEE_S16_S16(mllib_s16 block[64], const
mllib_s16 coeffs[64]);
```

DESCRIPTION The mllib_VideoIDCT_IEEE_S16_S16() function computes the inverse DCT (called IDCT) for the output IDCT block of data type mllib_s16 and input DCT coefficients of data type mllib_s16. This function is guaranteed to be IEEE-1180-compliant. The output of the IDCT routine is the difference between the current block and the reference block. The difference pixel can occupy nine bits and is represented as a 16-bit datum. The output must be added to the motion-compensated reference block in order to reconstruct the current block.

PARAMETERS The function takes the following arguments:

block Pointer to an 8x8 motion-compensated block that is the difference between the reference block and the current block. block need not be 8-byte aligned.

coeffs Pointer to the source DCT coefficients. coeffs need not be 8-byte aligned.

RETURN VALUES The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoIDCT8x8_S16_S16(3MLIB), mllib_VideoIDCT8x8_S16_S16_DC(3MLIB), mllib_VideoIDCT8x8_S16_S16_NA(3MLIB), mllib_VideoIDCT8x8_S16_S16_Q1(3MLIB), mllib_VideoIDCT8x8_U8_S16(3MLIB), mllib_VideoIDCT8x8_U8_S16_DC(3MLIB), mllib_VideoIDCT8x8_U8_S16_NA(3MLIB), mllib_VideoIDCT8x8_U8_S16_Q1(3MLIB), attributes(5)

mllib_VideoInterpAveX_U8_U8_16x16(3MLIB)

NAME	mllib_VideoInterpAveX_U8_U8_16x16, mllib_VideoInterpAveX_U8_U8_16x8, mllib_VideoInterpAveX_U8_U8_8x16, mllib_VideoInterpAveX_U8_U8_8x8, mllib_VideoInterpAveX_U8_U8_8x4 – half-pixel interpolation in the X direction and averaging for reference block								
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoInterpAveX_U8_U8_16x16(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride); mllib_status mllib_VideoInterpAveX_U8_U8_16x8(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride); mllib_status mllib_VideoInterpAveX_U8_U8_8x16(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride); mllib_status mllib_VideoInterpAveX_U8_U8_8x8(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride); mllib_status mllib_VideoInterpAveX_U8_U8_8x4(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride);</pre>								
DESCRIPTION	Each of these functions performs half-pixel interpolation in the X direction and averaging for a reference block of data type mllib_u8 and a current block of data type mllib_u8. The stride applies to both the input reference block and the current block.								
PARAMETERS	Each of the functions takes the following arguments: <table border="0" style="margin-left: 20px;"> <tr> <td style="padding-right: 20px;"><i>curr_block</i></td> <td>Pointer to the current block. <i>curr_block</i> must be 8-byte aligned.</td> </tr> <tr> <td><i>ref_block</i></td> <td>Pointer to the reference block.</td> </tr> <tr> <td><i>frm_stride</i></td> <td>Stride, in bytes, between adjacent rows in a frame in both the current block and the reference block. <i>frm_stride</i> must be a multiple of eight.</td> </tr> <tr> <td><i>fld_stride</i></td> <td>Stride, in bytes, between adjacent rows in a field in both the current block and reference block. <i>fld_stride</i> must be a multiple of eight.</td> </tr> </table>	<i>curr_block</i>	Pointer to the current block. <i>curr_block</i> must be 8-byte aligned.	<i>ref_block</i>	Pointer to the reference block.	<i>frm_stride</i>	Stride, in bytes, between adjacent rows in a frame in both the current block and the reference block. <i>frm_stride</i> must be a multiple of eight.	<i>fld_stride</i>	Stride, in bytes, between adjacent rows in a field in both the current block and reference block. <i>fld_stride</i> must be a multiple of eight.
<i>curr_block</i>	Pointer to the current block. <i>curr_block</i> must be 8-byte aligned.								
<i>ref_block</i>	Pointer to the reference block.								
<i>frm_stride</i>	Stride, in bytes, between adjacent rows in a frame in both the current block and the reference block. <i>frm_stride</i> must be a multiple of eight.								
<i>fld_stride</i>	Stride, in bytes, between adjacent rows in a field in both the current block and reference block. <i>fld_stride</i> must be a multiple of eight.								
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.								
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:								
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe		
ATTRIBUTE TYPE	ATTRIBUTE VALUE								
Interface Stability	Evolving								
MT-Level	MT-Safe								

mllib_VideoInterpAveX_U8_U8_16x16(3MLIB)

SEE ALSO | mllib_VideoAddBlock_U8_S16(3MLIB), mllib_VideoCopyRef_S16_U8(3MLIB),
mllib_VideoCopyRef_S16_U8_16x16(3MLIB),
mllib_VideoCopyRef_U8_U8(3MLIB),
mllib_VideoCopyRef_U8_U8_16x16(3MLIB),
mllib_VideoCopyRefAve_U8_U8(3MLIB),
mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB),
mllib_VideoH263OverlappedMC_S16_U8(3MLIB),
mllib_VideoH263OverlappedMC_U8_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8(3MLIB),
mllib_VideoInterpAveXY_U8_U8(3MLIB),
mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveY_U8_U8(3MLIB),
mllib_VideoInterpAveY_U8_U8_16x16(3MLIB),
mllib_VideoInterpX_S16_U8(3MLIB),
mllib_VideoInterpX_S16_U8_16x16(3MLIB),
mllib_VideoInterpX_U8_U8(3MLIB), mllib_VideoInterpXY_S16_U8(3MLIB),
mllib_VideoInterpXY_S16_U8_16x16(3MLIB),
mllib_VideoInterpXY_U8_U8(3MLIB),
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpY_S16_U8(3MLIB),
mllib_VideoInterpY_S16_U8_16x16(3MLIB),
mllib_VideoInterpY_U8_U8(3MLIB),
mllib_VideoInterpY_U8_U8_16x16(3MLIB),
mllib_VideoP64Decimate_U8_U8(3MLIB),
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),
attributes(5)

mllib_VideoInterpAveX_U8_U8(3MLIB)

NAME	mllib_VideoInterpAveX_U8_U8 – half-pixel interpolation in the X direction and averaging for reference block												
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoInterpAveX_U8_U8(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 width, mllib_s32 height, mllib_s32 frm_stride, mllib_s32 fld_stride);</pre>												
DESCRIPTION	The mllib_VideoInterpAveX_U8_U8() function performs half-pixel interpolation in the X direction and averaging for a reference block of data type mllib_u8 and a current block of data type mllib_u8. The stride applies to both the input reference block and the current block.												
PARAMETERS	<p>The function takes the following arguments:</p> <table border="0"> <tr> <td style="padding-right: 20px;"><i>curr_block</i></td> <td>Pointer to the current block. curr_block must be 8-byte aligned.</td> </tr> <tr> <td><i>ref_block</i></td> <td>Pointer to the reference block.</td> </tr> <tr> <td><i>width</i></td> <td>Width of the blocks.</td> </tr> <tr> <td><i>height</i></td> <td>Height of the blocks.</td> </tr> <tr> <td><i>frm_stride</i></td> <td>Stride, in bytes, between adjacent rows in a frame in both the current block and the reference block. frm_stride must be a multiple of eight.</td> </tr> <tr> <td><i>fld_stride</i></td> <td>Stride, in bytes, between adjacent rows in a field in both the current block and reference block. fld_stride must be a multiple of eight.</td> </tr> </table>	<i>curr_block</i>	Pointer to the current block. curr_block must be 8-byte aligned.	<i>ref_block</i>	Pointer to the reference block.	<i>width</i>	Width of the blocks.	<i>height</i>	Height of the blocks.	<i>frm_stride</i>	Stride, in bytes, between adjacent rows in a frame in both the current block and the reference block. frm_stride must be a multiple of eight.	<i>fld_stride</i>	Stride, in bytes, between adjacent rows in a field in both the current block and reference block. fld_stride must be a multiple of eight.
<i>curr_block</i>	Pointer to the current block. curr_block must be 8-byte aligned.												
<i>ref_block</i>	Pointer to the reference block.												
<i>width</i>	Width of the blocks.												
<i>height</i>	Height of the blocks.												
<i>frm_stride</i>	Stride, in bytes, between adjacent rows in a frame in both the current block and the reference block. frm_stride must be a multiple of eight.												
<i>fld_stride</i>	Stride, in bytes, between adjacent rows in a field in both the current block and reference block. fld_stride must be a multiple of eight.												
RETURN VALUES	The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.												
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:												
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe						
ATTRIBUTE TYPE	ATTRIBUTE VALUE												
Interface Stability	Evolving												
MT-Level	MT-Safe												
SEE ALSO	<pre>mllib_VideoAddBlock_U8_S16(3MLIB), mllib_VideoCopyRef_S16_U8(3MLIB), mllib_VideoCopyRef_S16_U8_16x16(3MLIB), mllib_VideoCopyRef_U8_U8(3MLIB), mllib_VideoCopyRef_U8_U8_16x16(3MLIB), mllib_VideoCopyRefAve_U8_U8(3MLIB), mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB), mllib_VideoH263OverlappedMC_S16_U8(3MLIB), mllib_VideoH263OverlappedMC_U8_U8(3MLIB), mllib_VideoInterpAveX_U8_U8_16x16(3MLIB),</pre>												

mllib_VideoInterpAveX_U8_U8(3MLIB)

```
mllib_VideoInterpAveXY_U8_U8(3MLIB),
mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveY_U8_U8(3MLIB),
mllib_VideoInterpAveY_U8_U8_16x16(3MLIB),
mllib_VideoInterpX_S16_U8(3MLIB),
mllib_VideoInterpX_S16_U8_16x16(3MLIB),
mllib_VideoInterpX_U8_U8(3MLIB), mllib_VideoInterpXY_S16_U8(3MLIB),
mllib_VideoInterpXY_S16_U8_16x16(3MLIB),
mllib_VideoInterpXY_U8_U8(3MLIB),
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpY_S16_U8(3MLIB),
mllib_VideoInterpY_S16_U8_16x16(3MLIB),
mllib_VideoInterpY_U8_U8(3MLIB),
mllib_VideoInterpY_U8_U8_16x16(3MLIB),
mllib_VideoP64Decimate_U8_U8(3MLIB),
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),
attributes(5)
```

mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB)

NAME	mllib_VideoInterpAveXY_U8_U8_16x16, mllib_VideoInterpAveXY_U8_U8_16x8, mllib_VideoInterpAveXY_U8_U8_8x16, mllib_VideoInterpAveXY_U8_U8_8x8, mllib_VideoInterpAveXY_U8_U8_8x4 – half-pixel interpolation in the X and Y directions and averaging for reference block						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoInterpAveXY_U8_U8_16x16(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride); mllib_status mllib_VideoInterpAveXY_U8_U8_16x8(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride); mllib_status mllib_VideoInterpAveXY_U8_U8_8x16(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride); mllib_status mllib_VideoInterpAveXY_U8_U8_8x8(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride); mllib_status mllib_VideoInterpAveXY_U8_U8_8x4(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride);</pre>						
DESCRIPTION	Each of these functions performs half-pixel interpolation in the X and Y directions and averaging for a reference block of data. In this mode, the motion-compensated reference block becomes the current block. Thus, the stride applies to both the input reference block and the current block.						
PARAMETERS	Each of the functions takes the following arguments: <p><i>curr_block</i> Pointer to the current block. <i>curr_block</i> must be 8-byte aligned.</p> <p><i>ref_block</i> Pointer to the reference block.</p> <p><i>frm_stride</i> Stride, in bytes, between adjacent rows in a frame in both the current block and the reference block. <i>frm_stride</i> must be a multiple of eight.</p> <p><i>fld_stride</i> Stride, in bytes, between adjacent rows in a field in both the current block and reference block. <i>fld_stride</i> must be a multiple of eight.</p>						
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:						
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						

mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB)

SEE ALSO | mllib_VideoAddBlock_U8_S16(3MLIB), mllib_VideoCopyRef_S16_U8(3MLIB),
mllib_VideoCopyRef_S16_U8_16x16(3MLIB),
mllib_VideoCopyRef_U8_U8(3MLIB),
mllib_VideoCopyRef_U8_U8_16x16(3MLIB),
mllib_VideoCopyRefAve_U8_U8(3MLIB),
mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB),
mllib_VideoH263OverlappedMC_S16_U8(3MLIB),
mllib_VideoH263OverlappedMC_U8_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveXY_U8_U8(3MLIB),
mllib_VideoInterpAveY_U8_U8(3MLIB),
mllib_VideoInterpAveY_U8_U8_16x16(3MLIB),
mllib_VideoInterpX_S16_U8(3MLIB),
mllib_VideoInterpX_S16_U8_16x16(3MLIB),
mllib_VideoInterpX_U8_U8(3MLIB), mllib_VideoInterpXY_S16_U8(3MLIB),
mllib_VideoInterpXY_S16_U8_16x16(3MLIB),
mllib_VideoInterpXY_U8_U8(3MLIB),
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpY_S16_U8(3MLIB),
mllib_VideoInterpY_S16_U8_16x16(3MLIB),
mllib_VideoInterpY_U8_U8(3MLIB),
mllib_VideoInterpY_U8_U8_16x16(3MLIB),
mllib_VideoP64Decimate_U8_U8(3MLIB),
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),
attributes(5)

mllib_VideoInterpAveXY_U8_U8(3MLIB)

NAME	mllib_VideoInterpAveXY_U8_U8 – half-pixel interpolation in the X and Y directions and averaging for reference block												
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoInterpAveXY_U8_U8(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 width, mllib_s32 height, mllib_s32 frm_stride, mllib_s32 fld_stride);</pre>												
DESCRIPTION	The mllib_VideoInterpAveXY_U8_U8 () function performs half-pixel interpolation in the X and Y directions and averaging for a reference block of data type mllib_u8 and a current block of data type mllib_u8. In this mode, the motion-compensated reference block becomes the current block. Thus, the stride applies to both the input reference block and the current block.												
PARAMETERS	<p>The function takes the following arguments:</p> <table border="0"> <tr> <td style="padding-right: 20px;"><i>curr_block</i></td> <td>Pointer to the current block. curr_block must be 8-byte aligned.</td> </tr> <tr> <td><i>ref_block</i></td> <td>Pointer to the reference block.</td> </tr> <tr> <td><i>width</i></td> <td>Width of the blocks.</td> </tr> <tr> <td><i>height</i></td> <td>Height of the blocks.</td> </tr> <tr> <td><i>frm_stride</i></td> <td>Stride, in bytes, between adjacent rows in a frame in both the current block and the reference block. frm_stride must be a multiple of eight.</td> </tr> <tr> <td><i>fld_stride</i></td> <td>Stride, in bytes, between adjacent rows in a field in both the current block and reference block. fld_stride must be a multiple of eight.</td> </tr> </table>	<i>curr_block</i>	Pointer to the current block. curr_block must be 8-byte aligned.	<i>ref_block</i>	Pointer to the reference block.	<i>width</i>	Width of the blocks.	<i>height</i>	Height of the blocks.	<i>frm_stride</i>	Stride, in bytes, between adjacent rows in a frame in both the current block and the reference block. frm_stride must be a multiple of eight.	<i>fld_stride</i>	Stride, in bytes, between adjacent rows in a field in both the current block and reference block. fld_stride must be a multiple of eight.
<i>curr_block</i>	Pointer to the current block. curr_block must be 8-byte aligned.												
<i>ref_block</i>	Pointer to the reference block.												
<i>width</i>	Width of the blocks.												
<i>height</i>	Height of the blocks.												
<i>frm_stride</i>	Stride, in bytes, between adjacent rows in a frame in both the current block and the reference block. frm_stride must be a multiple of eight.												
<i>fld_stride</i>	Stride, in bytes, between adjacent rows in a field in both the current block and reference block. fld_stride must be a multiple of eight.												
RETURN VALUES	The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.												
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:												
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe						
ATTRIBUTE TYPE	ATTRIBUTE VALUE												
Interface Stability	Evolving												
MT-Level	MT-Safe												
SEE ALSO	<pre>mllib_VideoAddBlock_U8_S16(3MLIB), mllib_VideoCopyRef_S16_U8(3MLIB), mllib_VideoCopyRef_S16_U8_16x16(3MLIB), mllib_VideoCopyRef_U8_U8(3MLIB), mllib_VideoCopyRef_U8_U8_16x16(3MLIB), mllib_VideoCopyRefAve_U8_U8(3MLIB), mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB), mllib_VideoH263OverlappedMC_S16_U8(3MLIB), mllib_VideoH263OverlappedMC_U8_U8(3MLIB),</pre>												

mllib_VideoInterpAveXY_U8_U8(3MLIB)

```
mllib_VideoInterpAveX_U8_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveY_U8_U8(3MLIB),
mllib_VideoInterpAveY_U8_U8_16x16(3MLIB),
mllib_VideoInterpX_S16_U8(3MLIB),
mllib_VideoInterpX_S16_U8_16x16(3MLIB),
mllib_VideoInterpX_U8_U8(3MLIB), mllib_VideoInterpXY_S16_U8(3MLIB),
mllib_VideoInterpXY_S16_U8_16x16(3MLIB),
mllib_VideoInterpXY_U8_U8(3MLIB),
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpY_S16_U8(3MLIB),
mllib_VideoInterpY_S16_U8_16x16(3MLIB),
mllib_VideoInterpY_U8_U8(3MLIB),
mllib_VideoInterpY_U8_U8_16x16(3MLIB),
mllib_VideoP64Decimate_U8_U8(3MLIB),
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),
attributes(5)
```

mllib_VideoInterpAveY_U8_U8_16x16(3MLIB)

NAME	mllib_VideoInterpAveY_U8_U8_16x16, mllib_VideoInterpAveY_U8_U8_16x8, mllib_VideoInterpAveY_U8_U8_8x16, mllib_VideoInterpAveY_U8_U8_8x8, mllib_VideoInterpAveY_U8_U8_8x4 – half-pixel interpolation in the Y direction and averaging for reference block								
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoInterpAveY_U8_U8_16x16(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride); mllib_status mllib_VideoInterpAveY_U8_U8_16x8(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride); mllib_status mllib_VideoInterpAveY_U8_U8_8x16(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride); mllib_status mllib_VideoInterpAveY_U8_U8_8x8(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride); mllib_status mllib_VideoInterpAveY_U8_U8_8x4(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride);</pre>								
DESCRIPTION	Each of these functions performs half-pixel interpolation in the Y direction and averaging for a reference block of data type mllib_u8 and a current block of data type mllib_u8. In this mode, the motion-compensated reference block becomes the current block. Thus, the stride applies to both the input reference block and the current block.								
PARAMETERS	Each of the functions takes the following arguments: <table border="0" style="margin-left: 20px;"> <tr> <td style="padding-right: 20px;"><i>curr_block</i></td> <td>Pointer to the current block. <i>curr_block</i> must be 8-byte aligned.</td> </tr> <tr> <td><i>ref_block</i></td> <td>Pointer to the reference block.</td> </tr> <tr> <td><i>frm_stride</i></td> <td>Stride, in bytes, between adjacent rows in a frame in both the current block and the reference block. <i>frm_stride</i> must be a multiple of eight.</td> </tr> <tr> <td><i>fld_stride</i></td> <td>Stride, in bytes, between adjacent rows in a field in both the current block and reference block. <i>fld_stride</i> must be a multiple of eight.</td> </tr> </table>	<i>curr_block</i>	Pointer to the current block. <i>curr_block</i> must be 8-byte aligned.	<i>ref_block</i>	Pointer to the reference block.	<i>frm_stride</i>	Stride, in bytes, between adjacent rows in a frame in both the current block and the reference block. <i>frm_stride</i> must be a multiple of eight.	<i>fld_stride</i>	Stride, in bytes, between adjacent rows in a field in both the current block and reference block. <i>fld_stride</i> must be a multiple of eight.
<i>curr_block</i>	Pointer to the current block. <i>curr_block</i> must be 8-byte aligned.								
<i>ref_block</i>	Pointer to the reference block.								
<i>frm_stride</i>	Stride, in bytes, between adjacent rows in a frame in both the current block and the reference block. <i>frm_stride</i> must be a multiple of eight.								
<i>fld_stride</i>	Stride, in bytes, between adjacent rows in a field in both the current block and reference block. <i>fld_stride</i> must be a multiple of eight.								
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.								
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:								
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe		
ATTRIBUTE TYPE	ATTRIBUTE VALUE								
Interface Stability	Evolving								
MT-Level	MT-Safe								

mllib_VideoInterpAveY_U8_U8_16x16(3MLIB)

SEE ALSO | mllib_VideoAddBlock_U8_S16(3MLIB), mllib_VideoCopyRef_S16_U8(3MLIB),
mllib_VideoCopyRef_S16_U8_16x16(3MLIB),
mllib_VideoCopyRef_U8_U8(3MLIB),
mllib_VideoCopyRef_U8_U8_16x16(3MLIB),
mllib_VideoCopyRefAve_U8_U8(3MLIB),
mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB),
mllib_VideoH263OverlappedMC_S16_U8(3MLIB),
mllib_VideoH263OverlappedMC_U8_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveXY_U8_U8(3MLIB),
mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveY_U8_U8(3MLIB), mllib_VideoInterpX_S16_U8(3MLIB),
mllib_VideoInterpX_S16_U8_16x16(3MLIB),
mllib_VideoInterpX_U8_U8(3MLIB), mllib_VideoInterpXY_S16_U8(3MLIB),
mllib_VideoInterpXY_S16_U8_16x16(3MLIB),
mllib_VideoInterpXY_U8_U8(3MLIB),
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpY_S16_U8(3MLIB),
mllib_VideoInterpY_S16_U8_16x16(3MLIB),
mllib_VideoInterpY_U8_U8(3MLIB),
mllib_VideoInterpY_U8_U8_16x16(3MLIB),
mllib_VideoP64Decimate_U8_U8(3MLIB),
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),
attributes(5)

mllib_VideoInterpAveY_U8_U8(3MLIB)

NAME	mllib_VideoInterpAveY_U8_U8 – half-pixel interpolation in the Y direction and averaging for reference block												
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_VideoInterpAveY_U8_U8(mllib_u8 *<i>curr_block</i>, const mllib_u8 *<i>ref_block</i>, mllib_s32 <i>width</i>, mllib_s32 <i>height</i>, mllib_s32 <i>frm_stride</i>, mllib_s32 <i>fld_stride</i>);</pre>												
DESCRIPTION	The mllib_VideoInterpAveY_U8_U8() function performs half-pixel interpolation in the Y direction and averaging for a reference block of data type mllib_u8 and a current block of data type mllib_u8. In this mode, the motion-compensated reference block becomes the current block. Thus, the stride applies to both the input reference block and the current block.												
PARAMETERS	The function takes the following arguments: <table><tr><td><i>curr_block</i></td><td>Pointer to the current block. <i>curr_block</i> must be 8-byte aligned.</td></tr><tr><td><i>ref_block</i></td><td>Pointer to the reference block.</td></tr><tr><td><i>width</i></td><td>Width of the blocks.</td></tr><tr><td><i>height</i></td><td>Height of the blocks.</td></tr><tr><td><i>frm_stride</i></td><td>Stride, in bytes, between adjacent rows in a frame in both the current block and the reference block. <i>frm_stride</i> must be a multiple of eight.</td></tr><tr><td><i>fld_stride</i></td><td>Stride, in bytes, between adjacent rows in a field in both the current block and reference block. <i>fld_stride</i> must be a multiple of eight.</td></tr></table>	<i>curr_block</i>	Pointer to the current block. <i>curr_block</i> must be 8-byte aligned.	<i>ref_block</i>	Pointer to the reference block.	<i>width</i>	Width of the blocks.	<i>height</i>	Height of the blocks.	<i>frm_stride</i>	Stride, in bytes, between adjacent rows in a frame in both the current block and the reference block. <i>frm_stride</i> must be a multiple of eight.	<i>fld_stride</i>	Stride, in bytes, between adjacent rows in a field in both the current block and reference block. <i>fld_stride</i> must be a multiple of eight.
<i>curr_block</i>	Pointer to the current block. <i>curr_block</i> must be 8-byte aligned.												
<i>ref_block</i>	Pointer to the reference block.												
<i>width</i>	Width of the blocks.												
<i>height</i>	Height of the blocks.												
<i>frm_stride</i>	Stride, in bytes, between adjacent rows in a frame in both the current block and the reference block. <i>frm_stride</i> must be a multiple of eight.												
<i>fld_stride</i>	Stride, in bytes, between adjacent rows in a field in both the current block and reference block. <i>fld_stride</i> must be a multiple of eight.												
RETURN VALUES	The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.												
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe						
ATTRIBUTE TYPE	ATTRIBUTE VALUE												
Interface Stability	Evolving												
MT-Level	MT-Safe												
SEE ALSO	mllib_VideoAddBlock_U8_S16(3MLIB), mllib_VideoCopyRef_S16_U8(3MLIB), mllib_VideoCopyRef_S16_U8_16x16(3MLIB), mllib_VideoCopyRef_U8_U8(3MLIB), mllib_VideoCopyRef_U8_U8_16x16(3MLIB), mllib_VideoCopyRefAve_U8_U8(3MLIB), mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB), mllib_VideoH263OverlappedMC_S16_U8(3MLIB), mllib_VideoH263OverlappedMC_U8_U8(3MLIB),												

mllib_VideoInterpAveY_U8_U8(3MLIB)

```
mllib_VideoInterpAveX_U8_U8(3MLIB),  
mllib_VideoInterpAveX_U8_U8_16x16(3MLIB),  
mllib_VideoInterpAveXY_U8_U8(3MLIB),  
mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB),  
mllib_VideoInterpAveY_U8_U8_16x16(3MLIB),  
mllib_VideoInterpX_S16_U8(3MLIB),  
mllib_VideoInterpX_S16_U8_16x16(3MLIB),  
mllib_VideoInterpX_U8_U8(3MLIB), mllib_VideoInterpXY_S16_U8(3MLIB),  
mllib_VideoInterpXY_S16_U8_16x16(3MLIB),  
mllib_VideoInterpXY_U8_U8(3MLIB),  
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),  
mllib_VideoInterpY_S16_U8(3MLIB),  
mllib_VideoInterpY_S16_U8_16x16(3MLIB),  
mllib_VideoInterpY_U8_U8(3MLIB),  
mllib_VideoInterpY_U8_U8_16x16(3MLIB),  
mllib_VideoP64Decimate_U8_U8(3MLIB),  
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),  
attributes(5)
```

mllib_VideoInterpX_S16_U8_16x16(3MLIB)

NAME | mllib_VideoInterpX_S16_U8_16x16, mllib_VideoInterpX_S16_U8_16x8, mllib_VideoInterpX_S16_U8_8x16, mllib_VideoInterpX_S16_U8_8x8, mllib_VideoInterpX_S16_U8_8x4 – half-pixel interpolation in the X direction

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoInterpX_S16_U8_16x16(mllib_s16 *mc_block,
      const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride);

mllib_status mllib_VideoInterpX_S16_U8_16x8(mllib_s16 *mc_block,
      const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride);

mllib_status mllib_VideoInterpX_S16_U8_8x16(mllib_s16 *mc_block,
      const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride);

mllib_status mllib_VideoInterpX_S16_U8_8x8(mllib_s16 *mc_block, const
      mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride);

mllib_status mllib_VideoInterpX_S16_U8_8x4(mllib_s16 *mc_block, const
      mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride);
```

DESCRIPTION | Each of these functions performs half-pixel interpolation in the X direction for a reference block of data type mllib_u8 and a current block of data type mllib_s16. In this mode, the output of this function must be added to the IDCT output to reconstruct the block in the current frame. Thus, the stride applies only to the input reference block.

PARAMETERS | Each of the functions takes the following arguments:

mc_block | Pointer to the motion-compensated reference block. *mc_block* must be 8-byte aligned.

ref_block | Pointer to the reference block.

frm_stride | Stride, in bytes, between adjacent rows in a frame in the reference block. *frm_stride* must be a multiple of eight.

fld_stride | Stride, in bytes, between adjacent rows in a field in the reference block. *fld_stride* must be a multiple of eight.

RETURN VALUES | Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mllib_VideoAddBlock_U8_S16(3MLIB), mllib_VideoCopyRef_S16_U8(3MLIB), mllib_VideoCopyRef_S16_U8_16x16(3MLIB), mllib_VideoCopyRef_U8_U8(3MLIB),

mllib_VideoInterpX_S16_U8_16x16(3MLIB)

```
mllib_VideoCopyRef_U8_U8_16x16(3MLIB),
mllib_VideoCopyRefAve_U8_U8(3MLIB),
mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB),
mllib_VideoH263OverlappedMC_S16_U8(3MLIB),
mllib_VideoH263OverlappedMC_U8_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveXY_U8_U8(3MLIB),
mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveY_U8_U8(3MLIB),
mllib_VideoInterpAveY_U8_U8_16x16(3MLIB),
mllib_VideoInterpX_S16_U8(3MLIB),
mllib_VideoInterpX_S16_U8_16x16(3MLIB),
mllib_VideoInterpX_U8_U8(3MLIB), mllib_VideoInterpXY_S16_U8(3MLIB),
mllib_VideoInterpXY_U8_U8(3MLIB),
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpY_S16_U8(3MLIB),
mllib_VideoInterpY_S16_U8_16x16(3MLIB),
mllib_VideoInterpY_U8_U8(3MLIB),
mllib_VideoInterpY_U8_U8_16x16(3MLIB),
mllib_VideoP64Decimate_U8_U8(3MLIB),
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),
attributes(5)
```

mllib_VideoInterpX_S16_U8(3MLIB)

NAME	mllib_VideoInterpX_S16_U8 – half-pixel interpolation in the X direction												
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoInterpX_S16_U8(mllib_s16 *mc_block, const mllib_u8 *ref_block, mllib_s32 width, mllib_s32 height, mllib_s32 frm_stride, mllib_s32 fld_stride);</pre>												
DESCRIPTION	The <code>mllib_VideoInterpX_S16_U8()</code> function performs half-pixel interpolation in the X direction for a reference block of data type <code>mllib_u8</code> and a current block of data type <code>mllib_s16</code> . In this mode, the output of this function must be added to the IDCT output to reconstruct the block in the current frame. Thus, the stride applies only to the input reference block.												
PARAMETERS	The function takes the following arguments: <table><tr><td><i>mc_block</i></td><td>Pointer to the motion-compensated reference block. <i>mc_block</i> must be 8-byte aligned.</td></tr><tr><td><i>ref_block</i></td><td>Pointer to the reference block.</td></tr><tr><td><i>width</i></td><td>Width of the blocks.</td></tr><tr><td><i>height</i></td><td>Height of the blocks.</td></tr><tr><td><i>frm_stride</i></td><td>Stride, in bytes, between adjacent rows in a frame in the reference block. <i>frm_stride</i> must be a multiple of eight.</td></tr><tr><td><i>fld_stride</i></td><td>Stride, in bytes, between adjacent rows in a field in the reference block. <i>fld_stride</i> must be a multiple of eight.</td></tr></table>	<i>mc_block</i>	Pointer to the motion-compensated reference block. <i>mc_block</i> must be 8-byte aligned.	<i>ref_block</i>	Pointer to the reference block.	<i>width</i>	Width of the blocks.	<i>height</i>	Height of the blocks.	<i>frm_stride</i>	Stride, in bytes, between adjacent rows in a frame in the reference block. <i>frm_stride</i> must be a multiple of eight.	<i>fld_stride</i>	Stride, in bytes, between adjacent rows in a field in the reference block. <i>fld_stride</i> must be a multiple of eight.
<i>mc_block</i>	Pointer to the motion-compensated reference block. <i>mc_block</i> must be 8-byte aligned.												
<i>ref_block</i>	Pointer to the reference block.												
<i>width</i>	Width of the blocks.												
<i>height</i>	Height of the blocks.												
<i>frm_stride</i>	Stride, in bytes, between adjacent rows in a frame in the reference block. <i>frm_stride</i> must be a multiple of eight.												
<i>fld_stride</i>	Stride, in bytes, between adjacent rows in a field in the reference block. <i>fld_stride</i> must be a multiple of eight.												
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .												
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe						
ATTRIBUTE TYPE	ATTRIBUTE VALUE												
Interface Stability	Evolving												
MT-Level	MT-Safe												
SEE ALSO	<code>mllib_VideoAddBlock_U8_S16(3MLIB)</code> , <code>mllib_VideoCopyRef_S16_U8(3MLIB)</code> , <code>mllib_VideoCopyRef_S16_U8_16x16(3MLIB)</code> , <code>mllib_VideoCopyRef_U8_U8(3MLIB)</code> , <code>mllib_VideoCopyRef_U8_U8_16x16(3MLIB)</code> , <code>mllib_VideoCopyRefAve_U8_U8(3MLIB)</code> , <code>mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB)</code> , <code>mllib_VideoH263OverlappedMC_S16_U8(3MLIB)</code> , <code>mllib_VideoH263OverlappedMC_U8_U8(3MLIB)</code> , <code>mllib_VideoInterpAveX_U8_U8(3MLIB)</code> , <code>mllib_VideoInterpAveX_U8_U8_16x16(3MLIB)</code> ,												

mllib_VideoInterpX_S16_U8(3MLIB)

```
mllib_VideoInterpAveXY_U8_U8(3MLIB),  
mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB),  
mllib_VideoInterpAveY_U8_U8(3MLIB),  
mllib_VideoInterpAveY_U8_U8_16x16(3MLIB),  
mllib_VideoInterpX_S16_U8_16x16(3MLIB),  
mllib_VideoInterpX_U8_U8(3MLIB), mllib_VideoInterpXY_S16_U8(3MLIB),  
mllib_VideoInterpXY_S16_U8_16x16(3MLIB),  
mllib_VideoInterpXY_U8_U8(3MLIB),  
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),  
mllib_VideoInterpY_S16_U8(3MLIB),  
mllib_VideoInterpY_S16_U8_16x16(3MLIB),  
mllib_VideoInterpY_U8_U8(3MLIB),  
mllib_VideoInterpY_U8_U8_16x16(3MLIB),  
mllib_VideoP64Decimate_U8_U8(3MLIB),  
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),  
attributes(5)
```

mllib_VideoInterpX_U8_U8_16x16(3MLIB)

NAME	mllib_VideoInterpX_U8_U8_16x16, mllib_VideoInterpX_U8_U8_16x8, mllib_VideoInterpX_U8_U8_8x16, mllib_VideoInterpX_U8_U8_8x8, mllib_VideoInterpX_U8_U8_8x4 – half-pixel interpolation in the X direction								
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoInterpX_U8_U8_16x16(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride); mllib_status mllib_VideoInterpX_U8_U8_16x8(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride); mllib_status mllib_VideoInterpX_U8_U8_8x16(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride); mllib_status mllib_VideoInterpX_U8_U8_8x8(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride); mllib_status mllib_VideoInterpX_U8_U8_8x4(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride);</pre>								
DESCRIPTION	Each of these functions performs half-pixel interpolation in the X direction for a reference block of data type mllib_u8 and a current block of data type mllib_u8. In this mode, the motion-compensated reference block becomes the current block. Thus, the stride applies to both the input reference block and the current block.								
PARAMETERS	Each of the functions takes the following arguments: <table border="0" style="margin-left: 20px;"> <tr> <td style="padding-right: 20px;"><i>curr_block</i></td> <td>Pointer to the current block. <i>curr_block</i> must be 8-byte aligned.</td> </tr> <tr> <td style="padding-right: 20px;"><i>ref_block</i></td> <td>Pointer to the reference block.</td> </tr> <tr> <td style="padding-right: 20px;"><i>frm_stride</i></td> <td>Stride, in bytes, between adjacent rows in a frame in both the current block and reference block. <i>frm_stride</i> must be a multiple of eight.</td> </tr> <tr> <td style="padding-right: 20px;"><i>fld_stride</i></td> <td>Stride, in bytes, between adjacent rows in a field in both the current block and reference block. <i>fld_stride</i> must be a multiple of eight.</td> </tr> </table>	<i>curr_block</i>	Pointer to the current block. <i>curr_block</i> must be 8-byte aligned.	<i>ref_block</i>	Pointer to the reference block.	<i>frm_stride</i>	Stride, in bytes, between adjacent rows in a frame in both the current block and reference block. <i>frm_stride</i> must be a multiple of eight.	<i>fld_stride</i>	Stride, in bytes, between adjacent rows in a field in both the current block and reference block. <i>fld_stride</i> must be a multiple of eight.
<i>curr_block</i>	Pointer to the current block. <i>curr_block</i> must be 8-byte aligned.								
<i>ref_block</i>	Pointer to the reference block.								
<i>frm_stride</i>	Stride, in bytes, between adjacent rows in a frame in both the current block and reference block. <i>frm_stride</i> must be a multiple of eight.								
<i>fld_stride</i>	Stride, in bytes, between adjacent rows in a field in both the current block and reference block. <i>fld_stride</i> must be a multiple of eight.								
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.								
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:								
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe		
ATTRIBUTE TYPE	ATTRIBUTE VALUE								
Interface Stability	Evolving								
MT-Level	MT-Safe								

mllib_VideoInterpX_U8_U8_16x16(3MLIB)

SEE ALSO | mllib_VideoAddBlock_U8_S16(3MLIB), mllib_VideoCopyRef_S16_U8(3MLIB),
mllib_VideoCopyRef_S16_U8_16x16(3MLIB),
mllib_VideoCopyRef_U8_U8(3MLIB),
mllib_VideoCopyRef_U8_U8_16x16(3MLIB),
mllib_VideoCopyRefAve_U8_U8(3MLIB),
mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB),
mllib_VideoH263OverlappedMC_S16_U8(3MLIB),
mllib_VideoH263OverlappedMC_U8_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveXY_U8_U8(3MLIB),
mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveY_U8_U8(3MLIB),
mllib_VideoInterpAveY_U8_U8_16x16(3MLIB),
mllib_VideoInterpX_S16_U8(3MLIB), mllib_VideoInterpX_U8_U8(3MLIB),
mllib_VideoInterpXY_S16_U8(3MLIB),
mllib_VideoInterpXY_S16_U8_16x16(3MLIB),
mllib_VideoInterpXY_U8_U8(3MLIB),
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpY_S16_U8(3MLIB),
mllib_VideoInterpY_S16_U8_16x16(3MLIB),
mllib_VideoInterpY_U8_U8(3MLIB),
mllib_VideoInterpY_U8_U8_16x16(3MLIB),
mllib_VideoP64Decimate_U8_U8(3MLIB),
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),
attributes(5)

mllib_VideoInterpX_U8_U8(3MLIB)

NAME	mllib_VideoInterpX_U8_U8 – half-pixel interpolation in the X direction												
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_VideoInterpX_U8_U8(mllib_u8 *<i>curr_block</i>, const mllib_u8 *<i>ref_block</i>, mllib_s32 <i>width</i>, mllib_s32 <i>height</i>, mllib_s32 <i>frm_stride</i>, mllib_s32 <i>fld_stride</i>);</pre>												
DESCRIPTION	The <code>mllib_VideoInterpX_U8_U8()</code> function performs half-pixel interpolation in the X direction for a reference block of data type <code>mllib_u8</code> and a current block of data type <code>mllib_u8</code> . In this mode, the motion-compensated reference block becomes the current block. Thus, the stride applies to both the input reference block and the current block.												
PARAMETERS	The function takes the following arguments: <table><tr><td><i>curr_block</i></td><td>Pointer to the current block. <i>curr_block</i> must be 8-byte aligned.</td></tr><tr><td><i>ref_block</i></td><td>Pointer to the reference block.</td></tr><tr><td><i>width</i></td><td>Width of the blocks.</td></tr><tr><td><i>height</i></td><td>Height of the blocks.</td></tr><tr><td><i>frm_stride</i></td><td>Stride, in bytes, between adjacent rows in a frame in both the current block and reference block. <i>frm_stride</i> must be a multiple of eight.</td></tr><tr><td><i>fld_stride</i></td><td>Stride, in bytes, between adjacent rows in a field in both the current block and reference block. <i>fld_stride</i> must be a multiple of eight.</td></tr></table>	<i>curr_block</i>	Pointer to the current block. <i>curr_block</i> must be 8-byte aligned.	<i>ref_block</i>	Pointer to the reference block.	<i>width</i>	Width of the blocks.	<i>height</i>	Height of the blocks.	<i>frm_stride</i>	Stride, in bytes, between adjacent rows in a frame in both the current block and reference block. <i>frm_stride</i> must be a multiple of eight.	<i>fld_stride</i>	Stride, in bytes, between adjacent rows in a field in both the current block and reference block. <i>fld_stride</i> must be a multiple of eight.
<i>curr_block</i>	Pointer to the current block. <i>curr_block</i> must be 8-byte aligned.												
<i>ref_block</i>	Pointer to the reference block.												
<i>width</i>	Width of the blocks.												
<i>height</i>	Height of the blocks.												
<i>frm_stride</i>	Stride, in bytes, between adjacent rows in a frame in both the current block and reference block. <i>frm_stride</i> must be a multiple of eight.												
<i>fld_stride</i>	Stride, in bytes, between adjacent rows in a field in both the current block and reference block. <i>fld_stride</i> must be a multiple of eight.												
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .												
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe						
ATTRIBUTE TYPE	ATTRIBUTE VALUE												
Interface Stability	Evolving												
MT-Level	MT-Safe												
SEE ALSO	<code>mllib_VideoAddBlock_U8_S16(3MLIB)</code> , <code>mllib_VideoCopyRef_S16_U8(3MLIB)</code> , <code>mllib_VideoCopyRef_S16_U8_16x16(3MLIB)</code> , <code>mllib_VideoCopyRef_U8_U8(3MLIB)</code> , <code>mllib_VideoCopyRef_U8_U8_16x16(3MLIB)</code> , <code>mllib_VideoCopyRefAve_U8_U8(3MLIB)</code> , <code>mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB)</code> , <code>mllib_VideoH263OverlappedMC_S16_U8(3MLIB)</code> , <code>mllib_VideoH263OverlappedMC_U8_U8(3MLIB)</code> , <code>mllib_VideoInterpAveX_U8_U8(3MLIB)</code> , <code>mllib_VideoInterpAveX_U8_U8_16x16(3MLIB)</code> ,												

mllib_VideoInterpX_U8_U8(3MLIB)

```
mllib_VideoInterpAveXY_U8_U8(3MLIB),
mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveY_U8_U8(3MLIB),
mllib_VideoInterpAveY_U8_U8_16x16(3MLIB),
mllib_VideoInterpX_S16_U8(3MLIB),
mllib_VideoInterpX_S16_U8_16x16(3MLIB),
mllib_VideoInterpXY_S16_U8(3MLIB),
mllib_VideoInterpXY_S16_U8_16x16(3MLIB),
mllib_VideoInterpXY_U8_U8(3MLIB),
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpY_S16_U8(3MLIB),
mllib_VideoInterpY_S16_U8_16x16(3MLIB),
mllib_VideoInterpY_U8_U8(3MLIB),
mllib_VideoInterpY_U8_U8_16x16(3MLIB),
mllib_VideoP64Decimate_U8_U8(3MLIB),
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),
attributes(5)
```

mllib_VideoInterpXY_S16_U8_16x16(3MLIB)

NAME	mllib_VideoInterpXY_S16_U8_16x16, mllib_VideoInterpXY_S16_U8_16x8, mllib_VideoInterpXY_S16_U8_8x16, mllib_VideoInterpXY_S16_U8_8x8, mllib_VideoInterpXY_S16_U8_8x4 – half-pixel interpolation in the X and Y directions for motion compensation								
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoInterpXY_S16_U8_16x16(mllib_s16 *mc_block, const mlib_u8 *ref_block, mlib_s32 frm_stride, mlib_s32 fld_stride); mllib_status mllib_VideoInterpXY_S16_U8_16x8(mllib_s16 *mc_block, const mlib_u8 *ref_block, mlib_s32 frm_stride, mlib_s32 fld_stride); mllib_status mllib_VideoInterpXY_S16_U8_8x16(mllib_s16 *mc_block, const mlib_u8 *ref_block, mlib_s32 frm_stride, mlib_s32 fld_stride); mllib_status mllib_VideoInterpXY_S16_U8_8x8(mllib_s16 *mc_block, const mlib_u8 *ref_block, mlib_s32 frm_stride, mlib_s32 fld_stride); mllib_status mllib_VideoInterpXY_S16_U8_8x4(mllib_s16 *mc_block, const mlib_u8 *ref_block, mlib_s32 frm_stride, mlib_s32 fld_stride);</pre>								
DESCRIPTION	Each of these functions performs half-pixel interpolation in the X and Y directions for a reference block of data type mlib_u8 and a current block of data type mlib_s16. In this mode, the output of this function must be added to the IDCT output to reconstruct the block in the current frame. Thus, the stride applies only to the input reference block.								
PARAMETERS	Each of the functions takes the following arguments: <table border="0" style="margin-left: 20px;"> <tr> <td style="padding-right: 20px;"><i>mc_block</i></td> <td>Pointer to the motion-compensated reference block. <i>mc_block</i> must be 8-byte aligned.</td> </tr> <tr> <td><i>ref_block</i></td> <td>Pointer to the reference block.</td> </tr> <tr> <td><i>frm_stride</i></td> <td>Stride, in bytes, between adjacent rows in a frame in the reference block. <i>frm_stride</i> must be a multiple of eight.</td> </tr> <tr> <td><i>fld_stride</i></td> <td>Stride, in bytes, between adjacent rows in a field in the reference block. <i>fld_stride</i> must be a multiple of eight.</td> </tr> </table>	<i>mc_block</i>	Pointer to the motion-compensated reference block. <i>mc_block</i> must be 8-byte aligned.	<i>ref_block</i>	Pointer to the reference block.	<i>frm_stride</i>	Stride, in bytes, between adjacent rows in a frame in the reference block. <i>frm_stride</i> must be a multiple of eight.	<i>fld_stride</i>	Stride, in bytes, between adjacent rows in a field in the reference block. <i>fld_stride</i> must be a multiple of eight.
<i>mc_block</i>	Pointer to the motion-compensated reference block. <i>mc_block</i> must be 8-byte aligned.								
<i>ref_block</i>	Pointer to the reference block.								
<i>frm_stride</i>	Stride, in bytes, between adjacent rows in a frame in the reference block. <i>frm_stride</i> must be a multiple of eight.								
<i>fld_stride</i>	Stride, in bytes, between adjacent rows in a field in the reference block. <i>fld_stride</i> must be a multiple of eight.								
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.								
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:								
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe		
ATTRIBUTE TYPE	ATTRIBUTE VALUE								
Interface Stability	Evolving								
MT-Level	MT-Safe								

mllib_VideoInterpXY_S16_U8_16x16(3MLIB)

SEE ALSO | mllib_VideoAddBlock_U8_S16(3MLIB), mllib_VideoCopyRef_S16_U8(3MLIB),
mllib_VideoCopyRef_S16_U8_16x16(3MLIB),
mllib_VideoCopyRef_U8_U8(3MLIB),
mllib_VideoCopyRef_U8_U8_16x16(3MLIB),
mllib_VideoCopyRefAve_U8_U8(3MLIB),
mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB),
mllib_VideoH263OverlappedMC_S16_U8(3MLIB),
mllib_VideoH263OverlappedMC_U8_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveXY_U8_U8(3MLIB),
mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveY_U8_U8(3MLIB),
mllib_VideoInterpAveY_U8_U8_16x16(3MLIB),
mllib_VideoInterpX_S16_U8(3MLIB),
mllib_VideoInterpX_S16_U8_16x16(3MLIB),
mllib_VideoInterpX_U8_U8(3MLIB), mllib_VideoInterpXY_S16_U8(3MLIB),
mllib_VideoInterpXY_U8_U8(3MLIB),
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpY_S16_U8(3MLIB),
mllib_VideoInterpY_S16_U8_16x16(3MLIB),
mllib_VideoInterpY_U8_U8(3MLIB),
mllib_VideoInterpY_U8_U8_16x16(3MLIB),
mllib_VideoP64Decimate_U8_U8(3MLIB),
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),
attributes(5)

mllib_VideoInterpXY_S16_U8(3MLIB)

NAME | mllib_VideoInterpXY_S16_U8 – half-pixel interpolation in the X and Y directions

SYNOPSIS |

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoInterpXY_S16_U8(mllib_s16 *mc_block, const
    mllib_u8 *ref_block, mllib_s32 width, mllib_s32 height, mllib_s32
    frm_stride, mllib_s32 fld_stride);
```

DESCRIPTION | The `mllib_VideoInterpXY_S16_U8()` function performs half-pixel interpolation in the X and Y directions for a reference block of data type `mllib_u8` and a current block of data type `mllib_s16`. In this mode, the output of this function must be added to the IDCT output to reconstruct the block in the current frame. Thus, the stride applies only to the input reference block.

PARAMETERS | The function takes the following arguments:

<i>mc_block</i>	Pointer to the motion-compensated reference block. <code>mc_block</code> must be 8-byte aligned.
<i>ref_block</i>	Pointer to the reference block.
<i>width</i>	Width of the blocks.
<i>height</i>	Height of the blocks.
<i>frm_stride</i>	Stride, in bytes, between adjacent rows in a frame in the reference block. <code>frm_stride</code> must be a multiple of eight.
<i>fld_stride</i>	Stride, in bytes, between adjacent rows in a field in the reference block. <code>fld_stride</code> must be a multiple of eight.

RETURN VALUES | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

ATTRIBUTES | See `attributes(5)` for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | `mllib_VideoAddBlock_U8_S16(3MLIB)`, `mllib_VideoCopyRef_S16_U8(3MLIB)`, `mllib_VideoCopyRef_S16_U8_16x16(3MLIB)`, `mllib_VideoCopyRef_U8_U8(3MLIB)`, `mllib_VideoCopyRef_U8_U8_16x16(3MLIB)`, `mllib_VideoCopyRefAve_U8_U8(3MLIB)`, `mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB)`, `mllib_VideoH263OverlappedMC_S16_U8(3MLIB)`, `mllib_VideoH263OverlappedMC_U8_U8(3MLIB)`, `mllib_VideoInterpAveX_U8_U8(3MLIB)`, `mllib_VideoInterpAveX_U8_U8_16x16(3MLIB)`,

mllib_VideoInterpXY_S16_U8(3MLIB)

```
mllib_VideoInterpAveXY_U8_U8(3MLIB),
mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveY_U8_U8(3MLIB),
mllib_VideoInterpAveY_U8_U8_16x16(3MLIB),
mllib_VideoInterpX_S16_U8(3MLIB),
mllib_VideoInterpX_S16_U8_16x16(3MLIB),
mllib_VideoInterpX_U8_U8(3MLIB),
mllib_VideoInterpXY_S16_U8_16x16(3MLIB),
mllib_VideoInterpXY_U8_U8(3MLIB),
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpY_S16_U8(3MLIB),
mllib_VideoInterpY_S16_U8_16x16(3MLIB),
mllib_VideoInterpY_U8_U8(3MLIB),
mllib_VideoInterpY_U8_U8_16x16(3MLIB),
mllib_VideoP64Decimate_U8_U8(3MLIB),
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),
attributes(5)
```

mllib_VideoInterpXY_U8_U8_16x16(3MLIB)

NAME	mllib_VideoInterpXY_U8_U8_16x16, mllib_VideoInterpXY_U8_U8_16x8, mllib_VideoInterpXY_U8_U8_8x16, mllib_VideoInterpXY_U8_U8_8x8, mllib_VideoInterpXY_U8_U8_8x4 – half-pixel interpolation in the X and Y directions								
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoInterpXY_U8_U8_16x16(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride); mllib_status mllib_VideoInterpXY_U8_U8_16x8(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride); mllib_status mllib_VideoInterpXY_U8_U8_8x16(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride); mllib_status mllib_VideoInterpXY_U8_U8_8x8(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride); mllib_status mllib_VideoInterpXY_U8_U8_8x4(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride);</pre>								
DESCRIPTION	Each of these functions performs half-pixel interpolation in the X and Y directions for a reference block of data type mllib_u8 and a current block of data type mllib_u8. In this mode, the motion-compensated reference block becomes the current block. Thus, the stride applies to both the input reference block and the current block.								
PARAMETERS	Each of the functions takes the following arguments: <table border="0" style="margin-left: 20px;"> <tr> <td style="padding-right: 20px;"><i>curr_block</i></td> <td>Pointer to the current block. <i>curr_block</i> must be 8-byte aligned.</td> </tr> <tr> <td style="padding-right: 20px;"><i>ref_block</i></td> <td>Pointer to the reference block.</td> </tr> <tr> <td style="padding-right: 20px;"><i>frm_stride</i></td> <td>Stride, in bytes, between adjacent rows in a frame in both the current block and reference block. <i>frm_stride</i> must be a multiple of eight.</td> </tr> <tr> <td style="padding-right: 20px;"><i>fld_stride</i></td> <td>Stride, in bytes, between adjacent rows in a field in both the current block and reference block. <i>fld_stride</i> must be a multiple of eight.</td> </tr> </table>	<i>curr_block</i>	Pointer to the current block. <i>curr_block</i> must be 8-byte aligned.	<i>ref_block</i>	Pointer to the reference block.	<i>frm_stride</i>	Stride, in bytes, between adjacent rows in a frame in both the current block and reference block. <i>frm_stride</i> must be a multiple of eight.	<i>fld_stride</i>	Stride, in bytes, between adjacent rows in a field in both the current block and reference block. <i>fld_stride</i> must be a multiple of eight.
<i>curr_block</i>	Pointer to the current block. <i>curr_block</i> must be 8-byte aligned.								
<i>ref_block</i>	Pointer to the reference block.								
<i>frm_stride</i>	Stride, in bytes, between adjacent rows in a frame in both the current block and reference block. <i>frm_stride</i> must be a multiple of eight.								
<i>fld_stride</i>	Stride, in bytes, between adjacent rows in a field in both the current block and reference block. <i>fld_stride</i> must be a multiple of eight.								
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.								
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:								
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe		
ATTRIBUTE TYPE	ATTRIBUTE VALUE								
Interface Stability	Evolving								
MT-Level	MT-Safe								

mllib_VideoInterpXY_U8_U8_16x16(3MLIB)

SEE ALSO | mllib_VideoAddBlock_U8_S16(3MLIB), mllib_VideoCopyRef_S16_U8(3MLIB),
mllib_VideoCopyRef_S16_U8_16x16(3MLIB),
mllib_VideoCopyRef_U8_U8(3MLIB),
mllib_VideoCopyRef_U8_U8_16x16(3MLIB),
mllib_VideoCopyRefAve_U8_U8(3MLIB),
mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB),
mllib_VideoH263OverlappedMC_S16_U8(3MLIB),
mllib_VideoH263OverlappedMC_U8_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveXY_U8_U8(3MLIB),
mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveY_U8_U8(3MLIB),
mllib_VideoInterpAveY_U8_U8_16x16(3MLIB),
mllib_VideoInterpX_S16_U8(3MLIB),
mllib_VideoInterpX_S16_U8_16x16(3MLIB),
mllib_VideoInterpX_U8_U8(3MLIB), mllib_VideoInterpXY_S16_U8(3MLIB),
mllib_VideoInterpXY_S16_U8_16x16(3MLIB),
mllib_VideoInterpXY_U8_U8(3MLIB), mllib_VideoInterpY_S16_U8(3MLIB),
mllib_VideoInterpY_S16_U8_16x16(3MLIB),
mllib_VideoInterpY_U8_U8(3MLIB),
mllib_VideoInterpY_U8_U8_16x16(3MLIB),
mllib_VideoP64Decimate_U8_U8(3MLIB),
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),
attributes(5)

mllib_VideoInterpXY_U8_U8(3MLIB)

NAME	mllib_VideoInterpXY_U8_U8 – half-pixel interpolation in the X and Y directions												
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoInterpXY_U8_U8(mllib_u8 *curr_block, const mllib_u8 *ref_block, mllib_s32 width, mllib_s32 height, mllib_s32 frm_stride, mllib_s32 fld_stride);</pre>												
DESCRIPTION	The <code>mllib_VideoInterpXY_U8_U8()</code> function performs half-pixel interpolation in the X and Y directions for a reference block of data type <code>mllib_u8</code> and a current block of data type <code>mllib_u8</code> . In this mode, the motion-compensated reference block becomes the current block. Thus, the stride applies to both the input reference block and the current block.												
PARAMETERS	The function takes the following arguments: <table><tr><td><i>curr_block</i></td><td>Pointer to the current block. <i>curr_block</i> must be 8-byte aligned.</td></tr><tr><td><i>ref_block</i></td><td>Pointer to the reference block.</td></tr><tr><td><i>width</i></td><td>Width of the blocks.</td></tr><tr><td><i>height</i></td><td>Height of the blocks.</td></tr><tr><td><i>frm_stride</i></td><td>Stride, in bytes, between adjacent rows in a frame in both the current block and reference block. <i>frm_stride</i> must be a multiple of eight.</td></tr><tr><td><i>fld_stride</i></td><td>Stride, in bytes, between adjacent rows in a field in both the current block and reference block. <i>fld_stride</i> must be a multiple of eight.</td></tr></table>	<i>curr_block</i>	Pointer to the current block. <i>curr_block</i> must be 8-byte aligned.	<i>ref_block</i>	Pointer to the reference block.	<i>width</i>	Width of the blocks.	<i>height</i>	Height of the blocks.	<i>frm_stride</i>	Stride, in bytes, between adjacent rows in a frame in both the current block and reference block. <i>frm_stride</i> must be a multiple of eight.	<i>fld_stride</i>	Stride, in bytes, between adjacent rows in a field in both the current block and reference block. <i>fld_stride</i> must be a multiple of eight.
<i>curr_block</i>	Pointer to the current block. <i>curr_block</i> must be 8-byte aligned.												
<i>ref_block</i>	Pointer to the reference block.												
<i>width</i>	Width of the blocks.												
<i>height</i>	Height of the blocks.												
<i>frm_stride</i>	Stride, in bytes, between adjacent rows in a frame in both the current block and reference block. <i>frm_stride</i> must be a multiple of eight.												
<i>fld_stride</i>	Stride, in bytes, between adjacent rows in a field in both the current block and reference block. <i>fld_stride</i> must be a multiple of eight.												
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .												
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe						
ATTRIBUTE TYPE	ATTRIBUTE VALUE												
Interface Stability	Evolving												
MT-Level	MT-Safe												
SEE ALSO	<code>mllib_VideoAddBlock_U8_S16(3MLIB)</code> , <code>mllib_VideoCopyRef_S16_U8(3MLIB)</code> , <code>mllib_VideoCopyRef_S16_U8_16x16(3MLIB)</code> , <code>mllib_VideoCopyRef_U8_U8(3MLIB)</code> , <code>mllib_VideoCopyRef_U8_U8_16x16(3MLIB)</code> , <code>mllib_VideoCopyRefAve_U8_U8(3MLIB)</code> , <code>mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB)</code> , <code>mllib_VideoH263OverlappedMC_S16_U8(3MLIB)</code> , <code>mllib_VideoH263OverlappedMC_U8_U8(3MLIB)</code> , <code>mllib_VideoInterpAveX_U8_U8(3MLIB)</code> ,												

mllib_VideoInterpXY_U8_U8(3MLIB)

```
mllib_VideoInterpAveX_U8_U8_16x16(3MLIB),  
mllib_VideoInterpAveXY_U8_U8(3MLIB),  
mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB),  
mllib_VideoInterpAveY_U8_U8(3MLIB),  
mllib_VideoInterpAveY_U8_U8_16x16(3MLIB),  
mllib_VideoInterpX_S16_U8(3MLIB),  
mllib_VideoInterpX_S16_U8_16x16(3MLIB),  
mllib_VideoInterpX_U8_U8(3MLIB), mllib_VideoInterpXY_S16_U8(3MLIB),  
mllib_VideoInterpXY_S16_U8_16x16(3MLIB),  
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),  
mllib_VideoInterpY_S16_U8(3MLIB),  
mllib_VideoInterpY_S16_U8_16x16(3MLIB),  
mllib_VideoInterpY_U8_U8(3MLIB),  
mllib_VideoInterpY_U8_U8_16x16(3MLIB),  
mllib_VideoP64Decimate_U8_U8(3MLIB),  
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),  
attributes(5)
```

mllib_VideoInterpX_Y_XY_U8_U8(3MLIB)

NAME	mllib_VideoInterpX_Y_XY_U8_U8 – half-pixel interpolation in both X and Y directions for replenishment mode						
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoInterpX_Y_XY_U8_U8(mllib_u8 *outputX, mllib_u8 *outputY, mllib_u8 *outputXY, const mllib_u8 *image, mllib_s32 stride, mllib_s32 width, mllib_s32 height);</pre>						
DESCRIPTION	The mllib_VideoInterpX_Y_XY_U8_U8() function performs half-pixel interpolation in both X and Y directions for the replenishment mode.						
PARAMETERS	The function takes the following arguments: <i>outputX</i> Pointer to the output of X-interpolation. <i>outputX</i> must be 8-byte aligned. <i>outputY</i> Pointer to the output of Y-interpolation. <i>outputY</i> must be 8-byte aligned. <i>outputXY</i> Pointer to the output of XY-interpolation. <i>outputXY</i> must be 8-byte aligned. <i>image</i> Pointer to the image data. <i>image</i> must be 8-byte aligned <i>stride</i> Stride, in bytes, between adjacent rows in the image. <i>stride</i> must be a multiple of eight. <i>width</i> Width of the image. <i>width</i> must be a multiple of eight. <i>height</i> Height of the image. <i>height</i> must be a multiple of two.						
RETURN VALUES	The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.						
ATTRIBUTES	See attributes(5) for descriptions of the following attributes: <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>	ATTRIBUTE TYPE	ATTRIBUTE VALUE	Interface Stability	Evolving	MT-Level	MT-Safe
ATTRIBUTE TYPE	ATTRIBUTE VALUE						
Interface Stability	Evolving						
MT-Level	MT-Safe						
SEE ALSO	mllib_VideoInterpX_U8_U8(3MLIB), mllib_VideoInterpXY_U8_U8(3MLIB), mllib_VideoInterpY_U8_U8(3MLIB), attributes(5)						

mllib_VideoInterpY_S16_U8_16x16(3MLIB)

NAME mllib_VideoInterpY_S16_U8_16x16, mllib_VideoInterpY_S16_U8_16x8,
mllib_VideoInterpY_S16_U8_8x16, mllib_VideoInterpY_S16_U8_8x8,
mllib_VideoInterpY_S16_U8_8x4 – half-pixel interpolation in the Y direction

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoInterpY_S16_U8_16x16(mllib_s16 *mc_block,
      const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride);

mllib_status mllib_VideoInterpY_S16_U8_16x8(mllib_s16 *mc_block,
      const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride);

mllib_status mllib_VideoInterpY_S16_U8_8x16(mllib_s16 *mc_block,
      const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride);

mllib_status mllib_VideoInterpY_S16_U8_8x8(mllib_s16 *mc_block, const
      mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride);

mllib_status mllib_VideoInterpY_S16_U8_8x4(mllib_s16 *mc_block, const
      mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride);
```

DESCRIPTION Each of these functions performs half-pixel interpolation in the Y direction for a reference block of data type mllib_u8 and a current block of data type mllib_s16. In this mode, the output of this function must be added to the IDCT output to reconstruct the block in the current frame. Thus, the stride applies only to the input reference block.

PARAMETERS Each of the functions takes the following arguments:

mc_block Pointer to the motion-compensated reference block. *mc_block* must be 8-byte aligned.

ref_block Pointer to the reference block.

frm_stride Stride, in bytes, between adjacent rows in a frame in the reference block. *frm_stride* must be a multiple of eight.

fld_stride Stride, in bytes, between adjacent rows in a field in the reference block. *fld_stride* must be a multiple of eight.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoAddBlock_U8_S16(3MLIB), mllib_VideoCopyRef_S16_U8(3MLIB),
mllib_VideoCopyRef_S16_U8_16x16(3MLIB),
mllib_VideoCopyRef_U8_U8(3MLIB),

mllib_VideoInterpY_S16_U8_16x16(3MLIB)

```
mllib_VideoCopyRef_U8_U8_16x16(3MLIB),
mllib_VideoCopyRefAve_U8_U8(3MLIB),
mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB),
mllib_VideoH263OverlappedMC_S16_U8(3MLIB),
mllib_VideoH263OverlappedMC_U8_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveXY_U8_U8(3MLIB),
mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveY_U8_U8(3MLIB),
mllib_VideoInterpAveY_U8_U8_16x16(3MLIB),
mllib_VideoInterpX_S16_U8(3MLIB),
mllib_VideoInterpX_S16_U8_16x16(3MLIB),
mllib_VideoInterpX_U8_U8(3MLIB), mllib_VideoInterpXY_S16_U8(3MLIB),
mllib_VideoInterpXY_S16_U8_16x16(3MLIB),
mllib_VideoInterpXY_U8_U8(3MLIB),
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpY_S16_U8(3MLIB), mllib_VideoInterpY_U8_U8(3MLIB),
mllib_VideoInterpY_U8_U8_16x16(3MLIB),
mllib_VideoP64Decimate_U8_U8(3MLIB),
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),
attributes(5)
```


NAME mllib_VideoInterpY_S16_U8 – half-pixel interpolation in the Y direction

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoInterpY_S16_U8(mllib_s16 *mc_block, const
    mllib_u8 *ref_block, mllib_s32 width, mllib_s32 height, mllib_s32
    frm_stride, mllib_s32 fld_stride);
```

DESCRIPTION The mllib_VideoInterpY_S16_U8() function performs half-pixel interpolation in the Y direction for a reference block of data type mllib_u8 and a current block of data type mllib_s16. In this mode, the output of this function must be added to the IDCT output to reconstruct the block in the current frame. Thus, the stride applies only to the input reference block.

PARAMETERS The function takes the following arguments:

mc_block Pointer to the motion-compensated reference block. mc_block must be 8-byte aligned.

ref_block Pointer to the reference block.

width Width of the blocks.

height Height of the blocks.

frm_stride Stride, in bytes, between adjacent rows in a frame in the reference block. frm_stride must be a multiple of eight.

fld_stride Stride, in bytes, between adjacent rows in a field in the reference block. fld_stride must be a multiple of eight.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoAddBlock_U8_S16(3MLIB), mllib_VideoCopyRef_S16_U8(3MLIB), mllib_VideoCopyRef_S16_U8_16x16(3MLIB), mllib_VideoCopyRef_U8_U8(3MLIB), mllib_VideoCopyRef_U8_U8_16x16(3MLIB), mllib_VideoCopyRefAve_U8_U8(3MLIB), mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB), mllib_VideoH263OverlappedMC_S16_U8(3MLIB), mllib_VideoH263OverlappedMC_U8_U8(3MLIB), mllib_VideoInterpAveX_U8_U8(3MLIB), mllib_VideoInterpAveX_U8_U8_16x16(3MLIB),

mllib_VideoInterpY_S16_U8(3MLIB)

```
mllib_VideoInterpAveXY_U8_U8(3MLIB),
mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveY_U8_U8(3MLIB),
mllib_VideoInterpAveY_U8_U8_16x16(3MLIB),
mllib_VideoInterpX_S16_U8(3MLIB),
mllib_VideoInterpX_S16_U8_16x16(3MLIB),
mllib_VideoInterpX_U8_U8(3MLIB), mllib_VideoInterpXY_S16_U8(3MLIB),
mllib_VideoInterpXY_S16_U8_16x16(3MLIB),
mllib_VideoInterpXY_U8_U8(3MLIB),
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpY_S16_U8_16x16(3MLIB),
mllib_VideoInterpY_U8_U8(3MLIB),
mllib_VideoInterpY_U8_U8_16x16(3MLIB),
mllib_VideoP64Decimate_U8_U8(3MLIB),
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),
attributes(5)
```

mllib_VideoInterpY_U8_U8_16x16(3MLIB)

NAME mllib_VideoInterpY_U8_U8_16x16, mllib_VideoInterpY_U8_U8_16x8,
mllib_VideoInterpY_U8_U8_8x16, mllib_VideoInterpY_U8_U8_8x8,
mllib_VideoInterpY_U8_U8_8x4 – half-pixel interpolation in the Y direction

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoInterpY_U8_U8_16x16(mllib_u8 *curr_block,
      const mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride);

mllib_status mllib_VideoInterpY_U8_U8_16x8(mllib_u8 *curr_block, const
      mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride);

mllib_status mllib_VideoInterpY_U8_U8_8x16(mllib_u8 *curr_block, const
      mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride);

mllib_status mllib_VideoInterpY_U8_U8_8x8(mllib_u8 *curr_block, const
      mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride);

mllib_status mllib_VideoInterpY_U8_U8_8x4(mllib_u8 *curr_block, const
      mllib_u8 *ref_block, mllib_s32 frm_stride, mllib_s32 fld_stride);
```

DESCRIPTION Each of these functions performs half-pixel interpolation in the Y direction for a reference block of data type mllib_u8 and a current block of data type mllib_u8. In this mode, the motion-compensated reference block becomes the current block. Thus, the stride applies to both the input reference block and the current block.

PARAMETERS Each of the functions takes the following arguments:

curr_block Pointer to the current block. *curr_block* must be 8-byte aligned.

ref_block Pointer to the reference block.

frm_stride Stride, in bytes, between adjacent rows in a frame in both the current block and reference block. *frm_stride* must be a multiple of eight.

fld_stride Stride, in bytes, between adjacent rows in a field in both the current block and reference block. *fld_stride* must be a multiple of eight.

RETURN VALUES Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

mllib_VideoInterpY_U8_U8_16x16(3MLIB)

SEE ALSO | mllib_VideoAddBlock_U8_S16(3MLIB), mllib_VideoCopyRef_S16_U8(3MLIB),
mllib_VideoCopyRef_S16_U8_16x16(3MLIB),
mllib_VideoCopyRef_U8_U8(3MLIB),
mllib_VideoCopyRef_U8_U8_16x16(3MLIB),
mllib_VideoCopyRefAve_U8_U8(3MLIB),
mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB),
mllib_VideoH263OverlappedMC_S16_U8(3MLIB),
mllib_VideoH263OverlappedMC_U8_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8(3MLIB),
mllib_VideoInterpAveX_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveXY_U8_U8(3MLIB),
mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpAveY_U8_U8(3MLIB),
mllib_VideoInterpAveY_U8_U8_16x16(3MLIB),
mllib_VideoInterpX_S16_U8(3MLIB),
mllib_VideoInterpX_S16_U8_16x16(3MLIB),
mllib_VideoInterpX_U8_U8(3MLIB), mllib_VideoInterpXY_S16_U8(3MLIB),
mllib_VideoInterpXY_S16_U8_16x16(3MLIB),
mllib_VideoInterpXY_U8_U8(3MLIB),
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),
mllib_VideoInterpY_S16_U8(3MLIB),
mllib_VideoInterpY_S16_U8_16x16(3MLIB),
mllib_VideoInterpY_U8_U8(3MLIB), mllib_VideoP64Decimate_U8_U8(3MLIB),
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),
attributes(5)

NAME mllib_VideoInterpY_U8_U8 – half-pixel interpolation in the Y direction

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoInterpY_U8_U8(mllib_u8 *curr_block, const
    mllib_u8 *ref_block, mllib_s32 width, mllib_s32 height, mllib_s32
    frm_stride, mllib_s32 fld_stride);
```

DESCRIPTION The mllib_VideoInterpY_U8_U8() function performs half-pixel interpolation in the Y direction for a reference block of data type mllib_u8 and a current block of data type mllib_u8. In this mode, the motion-compensated reference block becomes the current block. Thus, the stride applies to both the input reference block and the current block.

PARAMETERS The function takes the following arguments:

curr_block Pointer to the current block. curr_block must be 8-byte aligned.

ref_block Pointer to the reference block.

width Width of the blocks.

height Height of the blocks.

frm_stride Stride, in bytes, between adjacent rows in a frame in both the current block and reference block. frm_stride must be a multiple of eight.

fld_stride Stride, in bytes, between adjacent rows in a field in both the current block and reference block. fld_stride must be a multiple of eight.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoAddBlock_U8_S16(3MLIB), mllib_VideoCopyRef_S16_U8(3MLIB), mllib_VideoCopyRef_S16_U8_16x16(3MLIB), mllib_VideoCopyRef_U8_U8(3MLIB), mllib_VideoCopyRef_U8_U8_16x16(3MLIB), mllib_VideoCopyRefAve_U8_U8(3MLIB), mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB), mllib_VideoH263OverlappedMC_S16_U8(3MLIB), mllib_VideoH263OverlappedMC_U8_U8(3MLIB), mllib_VideoInterpAveX_U8_U8(3MLIB), mllib_VideoInterpAveX_U8_U8_16x16(3MLIB),

mllib_VideoInterpY_U8_U8(3MLIB)

```
mllib_VideoInterpAveXY_U8_U8(3MLIB),  
mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB),  
mllib_VideoInterpAveY_U8_U8(3MLIB),  
mllib_VideoInterpAveY_U8_U8_16x16(3MLIB),  
mllib_VideoInterpX_S16_U8(3MLIB),  
mllib_VideoInterpX_S16_U8_16x16(3MLIB),  
mllib_VideoInterpX_U8_U8(3MLIB), mllib_VideoInterpXY_S16_U8(3MLIB),  
mllib_VideoInterpXY_S16_U8_16x16(3MLIB),  
mllib_VideoInterpXY_U8_U8(3MLIB),  
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),  
mllib_VideoInterpY_S16_U8(3MLIB),  
mllib_VideoInterpY_S16_U8_16x16(3MLIB),  
mllib_VideoInterpY_U8_U8_16x16(3MLIB),  
mllib_VideoP64Decimate_U8_U8(3MLIB),  
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),  
attributes(5)
```

NAME mllib_VideoP64Decimate_U8_U8 – averages the source raster image over 2x2 blocks and writes the results to the destination raster image

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoP64Decimate_U8_U8(mllib_u8 *dst, const
      mllib_u8 *src, mllib_s32 width, mllib_s32 height, mllib_s32 dst_stride,
      mllib_s32 src_stride);
```

DESCRIPTION The mllib_VideoP64Decimate_U8_U8() function averages the source raster image over 2x2 blocks and writes the results to the destination raster image. This function is used when the remote side is only capable of QCIF and our scanned image is source to the encoder in CIF format.

PARAMETERS The function takes the following arguments:

dst Pointer to the destination raster image.

src Pointer to the source raster image.

width Width of the image.

height Height of the image.

dst_stride Stride, in bytes, between adjacent rows in the destination image.

src_stride Stride, in bytes, between adjacent rows in the source image.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoAddBlock_U8_S16(3MLIB), mllib_VideoCopyRef_S16_U8(3MLIB), mllib_VideoCopyRef_S16_U8_16x16(3MLIB), mllib_VideoCopyRef_U8_U8_16x16(3MLIB), mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB), mllib_VideoH263OverlappedMC_S16_U8(3MLIB), mllib_VideoH263OverlappedMC_U8_U8(3MLIB), mllib_VideoInterpAveX_U8_U8(3MLIB), mllib_VideoInterpAveX_U8_U8_16x16(3MLIB), mllib_VideoInterpAveXY_U8_U8(3MLIB), mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB), mllib_VideoInterpAveY_U8_U8(3MLIB), mllib_VideoInterpAveY_U8_U8_16x16(3MLIB), mllib_VideoInterpX_S16_U8(3MLIB),

mllib_VideoP64Decimate_U8_U8(3MLIB)

```
mllib_VideoInterpX_S16_U8_16x16(3MLIB),  
mllib_VideoInterpX_U8_U8(3MLIB), mllib_VideoInterpXY_S16_U8(3MLIB),  
mllib_VideoInterpXY_S16_U8_16x16(3MLIB),  
mllib_VideoInterpXY_U8_U8(3MLIB),  
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),  
mllib_VideoInterpY_S16_U8(3MLIB),  
mllib_VideoInterpY_S16_U8_16x16(3MLIB),  
mllib_VideoInterpY_U8_U8(3MLIB),  
mllib_VideoInterpY_U8_U8_16x16(3MLIB),  
mllib_VideoP64Decimate_U8_U8(3MLIB),  
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),  
mllib_VideoP64Loop_S16_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),  
attributes(5)
```


mlib_VideoP64Loop_S16_U8(3MLIB)

NAME | mlib_VideoP64Loop_S16_U8 – applies a 2-dimensional (2D) 3x3 spatial filter on the reference block

SYNOPSIS |

```
cc [ flag... ] file... -lmlib [ library... ]
#include <mlib.h>

mlib_status mlib_VideoP64Loop_S16_U8(mlib_s16 mc_block[64], const
mlib_u8 *ref_block, mlib_s32 stride);
```

DESCRIPTION | The mlib_VideoP64Loop_S16_U8() function applies a 2-dimensional (2D) 3x3 spatial filter on the reference block. The filter is separable into 1D horizontal and vertical functions, where the filter coefficients are 0.25, 0.5, 0.25, except at the block edges where the coefficients are 0, 1, 0. In this mode, the output must be added to the IDCT output to reconstruct the block in the current frame. Thus, the stride applies only to the input reference block. This function requires the motion-compensated block to be 8-bit aligned.

PARAMETERS | The function takes the following arguments:

mc_block | Pointer to the motion-compensated reference block.

ref_block | Pointer to the reference block.

stride | Stride, in bytes, between adjacent rows in the reference block.

RETURN VALUES | The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES | See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO | mlib_VideoAddBlock_U8_S16(3MLIB), mlib_VideoCopyRef_S16_U8(3MLIB), mlib_VideoCopyRef_S16_U8_16x16(3MLIB), mlib_VideoCopyRef_U8_U8_16x16(3MLIB), mlib_VideoCopyRefAve_U8_U8_16x16(3MLIB), mlib_VideoH263OverlappedMC_S16_U8(3MLIB), mlib_VideoH263OverlappedMC_U8_U8(3MLIB), mlib_VideoInterpAveX_U8_U8(3MLIB), mlib_VideoInterpAveX_U8_U8_16x16(3MLIB), mlib_VideoInterpAveXY_U8_U8(3MLIB), mlib_VideoInterpAveXY_U8_U8_16x16(3MLIB), mlib_VideoInterpAveY_U8_U8(3MLIB), mlib_VideoInterpAveY_U8_U8_16x16(3MLIB), mlib_VideoInterpX_S16_U8(3MLIB), mlib_VideoInterpX_S16_U8_16x16(3MLIB), mlib_VideoInterpX_U8_U8(3MLIB), mlib_VideoInterpXY_S16_U8(3MLIB), mlib_VideoInterpXY_S16_U8_16x16(3MLIB),

mllib_VideoP64Loop_S16_U8(3MLIB)

```
mllib_VideoInterpXY_U8_U8(3MLIB),  
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),  
mllib_VideoInterpY_S16_U8(3MLIB),  
mllib_VideoInterpY_S16_U8_16x16(3MLIB),  
mllib_VideoInterpY_U8_U8(3MLIB),  
mllib_VideoInterpY_U8_U8_16x16(3MLIB),  
mllib_VideoP64Decimate_U8_U8(3MLIB), mllib_VideoP64Loop_U8_U8(3MLIB),  
attributes(5)
```

NAME mllib_VideoP64Loop_U8_U8 – applies a 2-dimensional (2D) 3x3 spatial filter on the reference block

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoP64Loop_U8_U8(mllib_u8 *curr_block, const
    mllib_u8 *ref_block, mllib_s32 stride);
```

DESCRIPTION The mllib_VideoP64Loop_U8_U8() function applies a 2-dimensional (2D) 3x3 spatial filter on the reference block. The filter is separable into 1D horizontal and vertical functions, where the filter coefficients are 0.25, 0.5, 0.25, except at the block edges where the coefficients are 0, 1, 0. In this mode, the motion-compensated reference block becomes the current block. Thus, the stride applies to both the input reference block and the current block.

PARAMETERS The function takes the following arguments:

curr_block Pointer to the current block.

ref_block Pointer to the reference block.

stride Stride, in bytes, between adjacent rows in both the current block and the reference block.

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoAddBlock_U8_S16(3MLIB), mllib_VideoCopyRef_S16_U8(3MLIB), mllib_VideoCopyRef_S16_U8_16x16(3MLIB), mllib_VideoCopyRef_U8_U8_16x16(3MLIB), mllib_VideoCopyRefAve_U8_U8_16x16(3MLIB), mllib_VideoH263OverlappedMC_S16_U8(3MLIB), mllib_VideoH263OverlappedMC_U8_U8(3MLIB), mllib_VideoInterpAveX_U8_U8(3MLIB), mllib_VideoInterpAveX_U8_U8_16x16(3MLIB), mllib_VideoInterpAveXY_U8_U8(3MLIB), mllib_VideoInterpAveXY_U8_U8_16x16(3MLIB), mllib_VideoInterpAveY_U8_U8(3MLIB), mllib_VideoInterpAveY_U8_U8_16x16(3MLIB), mllib_VideoInterpX_S16_U8(3MLIB), mllib_VideoInterpX_S16_U8_16x16(3MLIB), mllib_VideoInterpX_U8_U8(3MLIB), mllib_VideoInterpXY_S16_U8(3MLIB), mllib_VideoInterpXY_S16_U8_16x16(3MLIB),

mllib_VideoP64Loop_U8_U8(3MLIB)

```
mllib_VideoInterpXY_U8_U8(3MLIB),  
mllib_VideoInterpXY_U8_U8_16x16(3MLIB),  
mllib_VideoInterpY_S16_U8(3MLIB),  
mllib_VideoInterpY_S16_U8_16x16(3MLIB),  
mllib_VideoInterpY_U8_U8(3MLIB),  
mllib_VideoInterpY_U8_U8_16x16(3MLIB),  
mllib_VideoP64Decimate_U8_U8(3MLIB),  
mllib_VideoP64Loop_S16_U8(3MLIB), attributes(5)
```

NAME mllib_VideoQuantizeInit_S16 – quantization of forward Discrete Cosine Transform (DCT) coefficients

SYNOPSIS

```
cc [ flag... ] file... -lmllib [ library... ]
#include <mllib.h>

mllib_status mllib_VideoQuantizeInit_S16(mllib_d64 dqtable[64], const
mllib_s16 iqtable[64]);
```

DESCRIPTION The mllib_VideoQuantizeInit_S16() function initializes the quantization table. The following equation is used:

$$dqtable[i] = 1.0 / iqtable[i]; \quad 0 \leq i < 64$$

PARAMETERS The function takes the following arguments:

dqtable Pointer to quantizer table coefficients.

iqtable Pointer to original quantizer table coefficients:

$$0 < iqtable[i] < 128$$

RETURN VALUES The function returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.

ATTRIBUTES See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO mllib_VideoDCT2x2_S16_S16(3MLIB), mllib_VideoDCT4x4_S16_S16(3MLIB), mllib_VideoDCT8x8_S16_S16(3MLIB), mllib_VideoDCT8x8_S16_S16_B12(3MLIB), mllib_VideoDCT8x8_S16_S16_NA(3MLIB), mllib_VideoDCT8x8_S16_U8(3MLIB), mllib_VideoDCT8x8_S16_U8_NA(3MLIB), mllib_VideoDCT16x16_S16_S16(3MLIB), mllib_VideoDCT16x16_S16_S16_B10(3MLIB), mllib_VideoDeQuantize_S16(3MLIB), mllib_VideoDeQuantizeInit_S16(3MLIB), mllib_VideoQuantize_S16(3MLIB), attributes(5)

mllib_VideoQuantize_S16(3MLIB)

NAME	mllib_VideoQuantize_S16 – quantization of forward Discrete Cosine Transform (DCT) coefficients
SYNOPSIS	<pre>cc [<i>flag...</i>] <i>file...</i> -lmllib [<i>library...</i>] #include <mllib.h> mllib_status mllib_VideoQuantize_S16(mllib_s16 <i>icoeffs</i>[64], const mllib_d64 <i>dqtable</i>[64]);</pre>
DESCRIPTION	<p>The <code>mllib_VideoQuantize_S16()</code> function performs quantization on DCT coefficients.</p> <p>The following equation is used:</p> $\text{icoeffs}[i] = \text{icoeffs}[i] * \text{dqtable}[i]; \quad 0 \leq i < 64$
PARAMETERS	<p>The function takes the following arguments:</p> <p><i>icoeffs</i> Pointer to the output DCT coefficients: -2048 < <code>icoeffs[i]</code> < 2048 Note that <code>icoeffs</code> must be 8-byte aligned.</p> <p><i>dqtable</i> Pointer to quantizer table coefficients.</p>
RETURN VALUES	The function returns <code>MLIB_SUCCESS</code> if successful. Otherwise it returns <code>MLIB_FAILURE</code> .
ATTRIBUTES	See <code>attributes(5)</code> for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_VideoDCT2x2_S16_S16(3MLIB)`, `mllib_VideoDCT4x4_S16_S16(3MLIB)`,
`mllib_VideoDCT8x8_S16_S16(3MLIB)`,
`mllib_VideoDCT8x8_S16_S16_B12(3MLIB)`,
`mllib_VideoDCT8x8_S16_S16_NA(3MLIB)`, `mllib_VideoDCT8x8_S16_U8(3MLIB)`,
`mllib_VideoDCT8x8_S16_U8_NA(3MLIB)`,
`mllib_VideoDCT16x16_S16_S16(3MLIB)`,
`mllib_VideoDCT16x16_S16_S16_B10(3MLIB)`,
`mllib_VideoDeQuantize_S16(3MLIB)`,
`mllib_VideoDeQuantizeInit_S16(3MLIB)`,
`mllib_VideoQuantizeInit_S16(3MLIB)`, `attributes(5)`

mllib_VideoReversibleColorRGB2YUV_U8_U8(3MLIB)

NAME	mllib_VideoReversibleColorRGB2YUV_U8_U8, mllib_VideoReversibleColorRGB2YUV_S16_U8, mllib_VideoReversibleColorRGB2YUV_S16_S16, mllib_VideoReversibleColorRGB2YUV_S32_S16 – reversible color space conversion for wavelet transformation
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoReversibleColorRGB2YUV_U8_U8(mllib_u8 *y, mllib_u8 *u, mllib_u8 *v, const mllib_u8 *r, const mllib_u8 *g, const mllib_u8 *b, mllib_s32 n, mllib_s32 depth); mllib_status mllib_VideoReversibleColorRGB2YUV_S16_U8(mllib_s16 *y, mllib_s16 *u, mllib_s16 *v, const mllib_u8 *r, const mllib_u8 *g, const mllib_u8 *b, mllib_s32 n, mllib_s32 depth); mllib_status mllib_VideoReversibleColorRGB2YUV_S16_S16(mllib_s16 *y, mllib_s16 *u, mllib_s16 *v, const mllib_s16 *r, const mllib_s16 *g, const mllib_s16 *b, mllib_s32 n, mllib_s32 depth); mllib_status mllib_VideoReversibleColorRGB2YUV_S32_S16(mllib_s32 *y, mllib_s32 *u, mllib_s32 *v, const mllib_s16 *r, const mllib_s16 *g, const mllib_s16 *b, mllib_s32 n, mllib_s32 depth);</pre>
DESCRIPTION	Each of the functions provides support to reversible wavelet transformation. It is for reversible color space conversion.
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <p><i>y</i> Pointer to destination Y component.</p> <p><i>u</i> Pointer to destination U component.</p> <p><i>v</i> Pointer to destination V component.</p> <p><i>r</i> Pointer to source R component.</p> <p><i>g</i> Pointer to source G component.</p> <p><i>b</i> Pointer to source B component.</p> <p><i>n</i> Length of data.</p> <p><i>depth</i> Number of bit planes required to store the original R, G, and B components.</p>
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:

`mllib_VideoReversibleColorRGB2YUV_U8_U8(3MLIB)`

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mllib_VideoReversibleColorYUV2RGB_U8_U8(3MLIB)`, `attributes(5)`

mllib_VideoReversibleColorYUV2RGB_U8_U8(3MLIB)

NAME	mllib_VideoReversibleColorYUV2RGB_U8_U8, mllib_VideoReversibleColorYUV2RGB_U8_S16, mllib_VideoReversibleColorYUV2RGB_S16_S16, mllib_VideoReversibleColorYUV2RGB_S16_S32 – reversible color space conversion for wavelet transformation
SYNOPSIS	<pre>cc [flag...] file... -lmllib [library...] #include <mllib.h> mllib_status mllib_VideoReversibleColorYUV2RGB_U8_U8(mllib_u8 *r, mllib_u8 *g, mllib_u8 *b, const mllib_u8 *y, const mllib_u8 *u, const mllib_u8 *v, mllib_s32 n, mllib_s32 depth); mllib_status mllib_VideoReversibleColorYUV2RGB_U8_S16(mllib_u8 *r, mllib_u8 *g, mllib_u8 *b, const mllib_s16 *y, const mllib_s16 *u, const mllib_s16 *v, mllib_s32 n, mllib_s32 depth); mllib_status mllib_VideoReversibleColorYUV2RGB_S16_S16(mllib_s16 *r, mllib_s16 *g, mllib_s16 *b, const mllib_s16 *y, const mllib_s16 *u, const mllib_s16 *v, mllib_s32 n, mllib_s32 depth); mllib_status mllib_VideoReversibleColorYUV2RGB_S16_S32(mllib_s16 *r, mllib_s16 *g, mllib_s16 *b, const mllib_s32 *y, const mllib_s32 *u, const mllib_s32 *v, mllib_s32 n, mllib_s32 depth);</pre>
DESCRIPTION	Each of the functions provides support to reversible wavelet transformation. It is for reversible color space conversion.
PARAMETERS	<p>Each of the functions takes the following arguments:</p> <p><i>r</i> Pointer to destination R component.</p> <p><i>g</i> Pointer to destination G component.</p> <p><i>b</i> Pointer to destination B component.</p> <p><i>y</i> Pointer to source Y component.</p> <p><i>u</i> Pointer to source U component.</p> <p><i>v</i> Pointer to source V component.</p> <p><i>n</i> Length of data.</p> <p><i>depth</i> Number of bit planes required to store the original R, G, and B components.</p>
RETURN VALUES	Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.
ATTRIBUTES	See attributes(5) for descriptions of the following attributes:

`mlib_VideoReversibleColorYUV2RGB_U8_U8(3MLIB)`

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	Evolving
MT-Level	MT-Safe

SEE ALSO `mlib_VideoReversibleColorRGB2YUV_U8_U8(3MLIB)`, `attributes(5)`

mllib_VideoSignMagnitudeConvert_S16(3MLIB)

- NAME** mllib_VideoSignMagnitudeConvert_S16 – wavelet transformation, sign-magnitude conversion
- SYNOPSIS**
- ```
cc [flag...] file... -lmllib [library...]
#include <mllib.h>

mllib_status mllib_VideoSignMagnitudeConvert_S16(mllib_s16 *srcdst,
mllib_s32 n);
```
- DESCRIPTION** The `mllib_VideoSignMagnitudeConvert_S16()` function converts data between standard 2s complement signed integer representation and sign-magnitude representation.
- PARAMETERS** The function takes the following arguments:
- dst* Pointer to destination data array.
- src* Pointer to source data array.
- srcdst* Pointer to source and destination data array.
- n* Array size.
- RETURN VALUES** The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.
- ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Evolving        |
| MT-Level            | MT-Safe         |

- SEE ALSO** `mllib_VideoSignMagnitudeConvert_S16_S16(3MLIB)`,  
`mllib_VideoSignMagnitudeConvert_S32(3MLIB)`,  
`mllib_VideoSignMagnitudeConvert_S32_S32(3MLIB)`, `attributes(5)`

## mllib\_VideoSignMagnitudeConvert\_S16\_S16(3MLIB)

**NAME** mllib\_VideoSignMagnitudeConvert\_S16\_S16 – wavelet transformation, sign-magnitude conversion

**SYNOPSIS**

```
cc [flag...] file... -lmllib [library...]
#include <mllib.h>

mllib_status mllib_VideoSignMagnitudeConvert_S16_S16(mllib_s16 *dst,
const mllib_s16 *src, mllib_s32 n);
```

**DESCRIPTION** The `mllib_VideoSignMagnitudeConvert_S16_S16()` function converts data between standard 2s complement signed integer representation and sign-magnitude representation.

**PARAMETERS** The function takes the following arguments:

|               |                                               |
|---------------|-----------------------------------------------|
| <i>dst</i>    | Pointer to destination data array.            |
| <i>src</i>    | Pointer to source data array.                 |
| <i>srcdst</i> | Pointer to source and destination data array. |
| <i>n</i>      | Array size.                                   |

**RETURN VALUES** The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

**ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Evolving        |
| MT-Level            | MT-Safe         |

**SEE ALSO** `mllib_VideoSignMagnitudeConvert_S16(3MLIB)`, `mllib_VideoSignMagnitudeConvert_S32(3MLIB)`, `mllib_VideoSignMagnitudeConvert_S32_S32(3MLIB)`, `attributes(5)`

## mllib\_VideoSignMagnitudeConvert\_S32(3MLIB)

**NAME** mllib\_VideoSignMagnitudeConvert\_S32 – wavelet transformation, sign-magnitude conversion

**SYNOPSIS**

```
cc [flag...] file... -lmllib [library...]
#include <mllib.h>

mllib_status mllib_VideoSignMagnitudeConvert_S32(mllib_s32 *srcdst,
mllib_s32 n);
```

**DESCRIPTION** The `mllib_VideoSignMagnitudeConvert_S32()` function converts data between standard 2s complement signed integer representation and sign-magnitude representation.

**PARAMETERS** The function takes the following arguments:

*dst* Pointer to destination data array.

*src* Pointer to source data array.

*srcdst* Pointer to source and destination data array.

*n* Array size.

**RETURN VALUES** The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

**ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Evolving        |
| MT-Level            | MT-Safe         |

**SEE ALSO** `mllib_VideoSignMagnitudeConvert_S16(3MLIB)`,  
`mllib_VideoSignMagnitudeConvert_S16_S16(3MLIB)`,  
`mllib_VideoSignMagnitudeConvert_S32_S32(3MLIB)`, `attributes(5)`

## mllib\_VideoSignMagnitudeConvert\_S32\_S32(3MLIB)

**NAME** mllib\_VideoSignMagnitudeConvert\_S32\_S32 – wavelet transformation, sign-magnitude conversion

**SYNOPSIS**

```
cc [flag...] file... -lmllib [library...]
#include <mllib.h>

mllib_status mllib_VideoSignMagnitudeConvert_S32_S32(mllib_s32 *dst,
const mllib_s32 *src, mllib_s32 n);
```

**DESCRIPTION** The `mllib_VideoSignMagnitudeConvert_S32_S32()` function converts data between standard 2s complement signed integer representation and sign-magnitude representation.

**PARAMETERS** The function takes the following arguments:

|               |                                               |
|---------------|-----------------------------------------------|
| <i>dst</i>    | Pointer to destination data array.            |
| <i>src</i>    | Pointer to source data array.                 |
| <i>srcdst</i> | Pointer to source and destination data array. |
| <i>n</i>      | Array size.                                   |

**RETURN VALUES** The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

**ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Evolving        |
| MT-Level            | MT-Safe         |

**SEE ALSO** `mllib_VideoSignMagnitudeConvert_S16(3MLIB)`,  
`mllib_VideoSignMagnitudeConvert_S16_S16(3MLIB)`,  
`mllib_VideoSignMagnitudeConvert_S32(3MLIB)`, `attributes(5)`

**NAME** mllib\_VideoSumAbsDiff – motion estimation

**SYNOPSIS**

```
cc [flag...] file... -lmllib [library...]
#include <mllib.h>

mllib_s32 mllib_VideoSumAbsDiff(mllib_u8 *curr_block, const mllib_u8
 *ref_block, mllib_s32 width, mllib_s32 height, mllib_s32 stride);
```

**DESCRIPTION** The `mllib_VideoSumAbsDiff()` function computes the sum of absolute differences between the pixels in the current block and the corresponding pixels in the reference block.

Both the current block and the reference block belong to frames with the same dimension. (The stride is applicable to both.) Motion estimation computes the sum of the absolute differences between the current block and reference blocks at different locations in the reference frame, choosing the best fit (least sum of absolute difference) to calculate the motion vector.

**PARAMETERS** The function takes the following arguments:

*curr\_block* Pointer to the current block. *curr\_block* must be 8-byte aligned.

*ref\_block* Pointer to the reference block.

*width* Width of the block.

*height* Height of the block.

*stride* Stride, in bytes, between adjacent rows in a block. *stride* must be a multiple of eight.

**RETURN VALUES** The function returns a value of type `mllib_s32`.

**ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Evolving        |
| MT-Level            | MT-Safe         |

**SEE ALSO** `attributes(5)`

## mllib\_VideoUpSample420(3MLIB)

| <b>NAME</b>          | mllib_VideoUpSample420 – up sampling rate conversion in JFIF                                                                                                                                                                                                                                                                                                                                                                                                                                     |                |                 |                     |          |          |         |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|---------------------|----------|----------|---------|
| <b>SYNOPSIS</b>      | <pre>cc [ flag... ] file... -lmllib [ library... ] #include &lt;mllib.h&gt;  mllib_status mllib_VideoUpSample420(mllib_u8 *dst0, mllib_u8 *dst1,     const mllib_u8 *src0, const mllib_u8 *src1, const mllib_u8 *src2,     mllib_s32 n);</pre>                                                                                                                                                                                                                                                   |                |                 |                     |          |          |         |
| <b>DESCRIPTION</b>   | The mllib_VideoUpSample420() function performs up sampling rate conversion used in JPEG File Interchange Format (JFIF).                                                                                                                                                                                                                                                                                                                                                                          |                |                 |                     |          |          |         |
| <b>PARAMETERS</b>    | The function takes the following arguments:<br><i>dst0</i> Pointer to upper destination row. dst0 must be 8-byte aligned.<br><i>dst1</i> Pointer to lower destination row. dst1 must be 8-byte aligned.<br><i>src0</i> Pointer to upper source row. src0 must be 8-byte aligned.<br><i>src1</i> Pointer to middle source row. src1 must be 8-byte aligned.<br><i>src2</i> Pointer to lower source row. src2 must be 8-byte aligned.<br><i>n</i> Length of source rows. n must be greater than 1. |                |                 |                     |          |          |         |
| <b>RETURN VALUES</b> | The function returns MLLIB_SUCCESS if successful. Otherwise it returns MLLIB_FAILURE.                                                                                                                                                                                                                                                                                                                                                                                                            |                |                 |                     |          |          |         |
| <b>ATTRIBUTES</b>    | See attributes(5) for descriptions of the following attributes:<br><table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>                                                                                                                                                                                                              | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Evolving | MT-Level | MT-Safe |
| ATTRIBUTE TYPE       | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |                |                 |                     |          |          |         |
| Interface Stability  | Evolving                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |                |                 |                     |          |          |         |
| MT-Level             | MT-Safe                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                |                 |                     |          |          |         |
| <b>SEE ALSO</b>      | mllib_VideoDownSample420(3MLIB), mllib_VideoDownSample420_S16(3MLIB), mllib_VideoDownSample422(3MLIB), mllib_VideoDownSample422_S16(3MLIB), mllib_VideoUpSample420_Nearest(3MLIB), mllib_VideoUpSample420_Nearest_S16(3MLIB), mllib_VideoUpSample420_S16(3MLIB), mllib_VideoUpSample422(3MLIB), mllib_VideoUpSample422_S16(3MLIB), mllib_VideoUpSample422_Nearest(3MLIB), mllib_VideoUpSample422_Nearest_S16(3MLIB), attributes(5)                                                               |                |                 |                     |          |          |         |



mlib\_VideoUpSample420\_Nearest(3MLIB)

**NAME** | mlib\_VideoUpSample420\_Nearest – up sampling rate conversion in JFIF

**SYNOPSIS** | 

```
cc [flag...] file... -lmlib [library...]
#include <mlib.h>

mlib_status mlib_VideoUpSample420_Nearest(mlib_u8 *dst0, mlib_u8
 *dst1, const mlib_u8 *src, mlib_s32 n);
```

**DESCRIPTION** | The mlib\_VideoUpSample420\_Nearest () function performs up sampling rate conversion used in JPEG File Interchange Format (JFIF).

**PARAMETERS** | The function takes the following arguments:

*dst0* | Pointer to upper destination row. dst0 must be 8-byte aligned.

*dst1* | Pointer to lower destination row. dst1 must be 8-byte aligned.

*src* | Pointer to source row. src must be 8-byte aligned.

*n* | Length of source rows. n must be greater than 1.

**RETURN VALUES** | The function returns MLIB\_SUCCESS if successful. Otherwise it returns MLIB\_FAILURE.

**ATTRIBUTES** | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Evolving        |
| MT-Level            | MT-Safe         |

**SEE ALSO** | mlib\_VideoDownSample420(3MLIB), mlib\_VideoDownSample420\_S16(3MLIB), mlib\_VideoDownSample422(3MLIB), mlib\_VideoDownSample422\_S16(3MLIB), mlib\_VideoUpSample420(3MLIB), mlib\_VideoUpSample420\_Nearest\_S16(3MLIB), mlib\_VideoUpSample420\_S16(3MLIB), mlib\_VideoUpSample422(3MLIB), mlib\_VideoUpSample422\_S16(3MLIB), mlib\_VideoUpSample422\_Nearest(3MLIB), mlib\_VideoUpSample422\_Nearest\_S16(3MLIB), attributes(5)

## mllib\_VideoUpSample420\_Nearest\_S16(3MLIB)

**NAME** | mllib\_VideoUpSample420\_Nearest\_S16 – up sampling rate conversion in JFIF

**SYNOPSIS** | 

```
cc [flag...] file... -lmllib [library...]
#include <mllib.h>

mllib_status mllib_VideoUpSample420_Nearest_S16(mllib_s16 *dst0,
 mllib_s16 *dst1, const mllib_s16 *src, mllib_s32 n);
```

**DESCRIPTION** | The mllib\_VideoUpSample420\_Nearest\_S16() function performs up sampling rate conversion used in JPEG File Interchange Format (JFIF).

**PARAMETERS** | The function takes the following arguments:

*dst0* | Pointer to upper destination row. dst0 must be 8-byte aligned.

*dst1* | Pointer to lower destination row. dst1 must be 8-byte aligned.

*src* | Pointer to source row. src must be 8-byte aligned.

*n* | Length of source rows. n must be greater than 1.

**RETURN VALUES** | The function returns MLLIB\_SUCCESS if successful. Otherwise it returns MLLIB\_FAILURE.

**ATTRIBUTES** | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Evolving        |
| MT-Level            | MT-Safe         |

**SEE ALSO** | mllib\_VideoDownSample420(3MLIB), mllib\_VideoDownSample420\_S16(3MLIB), mllib\_VideoDownSample422(3MLIB), mllib\_VideoDownSample422\_S16(3MLIB), mllib\_VideoUpSample420(3MLIB), mllib\_VideoUpSample420\_Nearest(3MLIB), mllib\_VideoUpSample420\_S16(3MLIB), mllib\_VideoUpSample422(3MLIB), mllib\_VideoUpSample422\_S16(3MLIB), mllib\_VideoUpSample422\_Nearest(3MLIB), mllib\_VideoUpSample422\_Nearest\_S16(3MLIB), attributes(5)

**NAME** | mllib\_VideoUpSample420\_S16 – up sampling rate conversion in JFIF

**SYNOPSIS** | 

```
cc [flag...] file... -lmllib [library...]
#include <mllib.h>

mllib_status mllib_VideoUpSample420_S16(mllib_s16 *dst0, mllib_s16
 *dst1, const mllib_s16 *src0, const mllib_s16 *src1, const mllib_s16
 *src2, mllib_s32 n);
```

**DESCRIPTION** | The mllib\_VideoUpSample420\_S16() function performs up sampling rate conversion used in JPEG File Interchange Format (JFIF).

**PARAMETERS** | The function takes the following arguments:

*dst0* | Pointer to upper destination row. *dst0* must be 8-byte aligned.

*dst1* | Pointer to lower destination row. *dst1* must be 8-byte aligned.

*src0* | Pointer to upper source row. *src0* must be 8-byte aligned.

*src1* | Pointer to middle source row. *src1* must be 8-byte aligned.

*src2* | Pointer to lower source row. *src2* must be 8-byte aligned.

*n* | Length of source rows. *n* must be greater than 1.

**RETURN VALUES** | The function returns MLIB\_SUCCESS if successful. Otherwise it returns MLIB\_FAILURE.

**ATTRIBUTES** | See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Evolving        |
| MT-Level            | MT-Safe         |

**SEE ALSO** | mllib\_VideoDownSample420(3MLIB), mllib\_VideoDownSample420\_S16(3MLIB), mllib\_VideoDownSample422(3MLIB), mllib\_VideoDownSample422\_S16(3MLIB), mllib\_VideoUpSample420(3MLIB), mllib\_VideoUpSample420\_Nearest(3MLIB), mllib\_VideoUpSample420\_Nearest\_S16(3MLIB), mllib\_VideoUpSample422(3MLIB), mllib\_VideoUpSample422\_S16(3MLIB), mllib\_VideoUpSample422\_Nearest(3MLIB), mllib\_VideoUpSample422\_Nearest\_S16(3MLIB), attributes(5)

## mllib\_VideoUpSample422(3MLIB)

**NAME** mllib\_VideoUpSample422 – up sampling rate conversion in JFIF

**SYNOPSIS**

```
cc [flag...] file... -lmllib [library...]
#include <mllib.h>

mllib_status mllib_VideoUpSample422(mllib_u8 *dst, const mllib_u8 *src,
mllib_s32 n);
```

**DESCRIPTION** The mllib\_VideoUpSample422() function performs up sampling rate conversion used in JPEG File Interchange Format (JFIF).

**PARAMETERS** The function takes the following arguments:

*dst* Pointer to destination row. dst must be 8-byte aligned.

*src* Pointer to source row. src must be 8-byte aligned.

*n* Length of source rows. n must be greater than 1.

**RETURN VALUES** The function returns MLLIB\_SUCCESS if successful. Otherwise it returns MLLIB\_FAILURE.

**ATTRIBUTES** See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Evolving        |
| MT-Level            | MT-Safe         |

**SEE ALSO** mllib\_VideoDownSample420(3MLIB), mllib\_VideoDownSample420\_S16(3MLIB), mllib\_VideoDownSample422(3MLIB), mllib\_VideoDownSample422\_S16(3MLIB), mllib\_VideoUpSample420(3MLIB), mllib\_VideoUpSample420\_Nearest(3MLIB), mllib\_VideoUpSample420\_Nearest\_S16(3MLIB), mllib\_VideoUpSample420\_S16(3MLIB), mllib\_VideoUpSample422\_S16(3MLIB), mllib\_VideoUpSample422\_Nearest(3MLIB), mllib\_VideoUpSample422\_Nearest\_S16(3MLIB), attributes(5)

mlib\_VideoUpSample422\_Nearest(3MLIB)

**NAME** | mlib\_VideoUpSample422\_Nearest – up sampling rate conversion in JFIF

**SYNOPSIS** | 

```
cc [flag...] file... -lmlib [library...]
#include <mlib.h>

mlib_status mlib_VideoUpSample422_Nearest(mlib_u8 *dst, const
mlib_u8 *src, mlib_s32 n);
```

**DESCRIPTION** | The `mlib_VideoUpSample422_Nearest()` function performs up sampling rate conversion used in JPEG File Interchange Format (JFIF).

**PARAMETERS** | The function takes the following arguments:

*dst* | Pointer to destination row. *dst* must be 8-byte aligned.

*src* | Pointer to source row. *src* must be 8-byte aligned.

*n* | Length of source rows. *n* must be greater than 1.

**RETURN VALUES** | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

**ATTRIBUTES** | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Evolving        |
| MT-Level            | MT-Safe         |

**SEE ALSO** | `mlib_VideoDownSample420(3MLIB)`, `mlib_VideoDownSample420_S16(3MLIB)`, `mlib_VideoDownSample422(3MLIB)`, `mlib_VideoDownSample422_S16(3MLIB)`, `mlib_VideoUpSample420(3MLIB)`, `mlib_VideoUpSample420_Nearest(3MLIB)`, `mlib_VideoUpSample420_Nearest_S16(3MLIB)`, `mlib_VideoUpSample420_S16(3MLIB)`, `mlib_VideoUpSample422(3MLIB)`, `mlib_VideoUpSample422_S16(3MLIB)`, `mlib_VideoUpSample422_Nearest_S16(3MLIB)`, `attributes(5)`

## mllib\_VideoUpSample422\_Nearest\_S16(3MLIB)

**NAME** | mllib\_VideoUpSample422\_Nearest\_S16 – up sampling rate conversion in JFIF

**SYNOPSIS** | 

```
cc [flag...] file... -lmllib [library...]
#include <mllib.h>

mllib_status mllib_VideoUpSample422_Nearest_S16(mllib_s16 *dst,
const mllib_s16 *src, mllib_s32 n);
```

**DESCRIPTION** | The `mllib_VideoUpSample422_Nearest_S16()` function performs up sampling rate conversion used in JPEG File Interchange Format (JFIF).

**PARAMETERS** | The function takes the following arguments:

|            |                                                                |
|------------|----------------------------------------------------------------|
| <i>dst</i> | Pointer to destination row. <i>dst</i> must be 8-byte aligned. |
| <i>src</i> | Pointer to source row. <i>src</i> must be 8-byte aligned.      |
| <i>n</i>   | Length of source rows. <i>n</i> must be greater than 1.        |

**RETURN VALUES** | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

**ATTRIBUTES** | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Evolving        |
| MT-Level            | MT-Safe         |

**SEE ALSO** | `mllib_VideoDownSample420(3MLIB)`, `mllib_VideoDownSample420_S16(3MLIB)`, `mllib_VideoDownSample422(3MLIB)`, `mllib_VideoDownSample422_S16(3MLIB)`, `mllib_VideoUpSample420(3MLIB)`, `mllib_VideoUpSample420_Nearest(3MLIB)`, `mllib_VideoUpSample420_Nearest_S16(3MLIB)`, `mllib_VideoUpSample420_S16(3MLIB)`, `mllib_VideoUpSample422(3MLIB)`, `mllib_VideoUpSample422_S16(3MLIB)`, `mllib_VideoUpSample422_Nearest(3MLIB)`, `attributes(5)`

mlib\_VideoUpSample422\_S16(3MLIB)

**NAME** | mlib\_VideoUpSample422\_S16 – up sampling rate conversion in JFIF

**SYNOPSIS** | `cc [ flag... ] file... -lmlib [ library... ]`  
`#include <mlib.h>`

`mlib_status mlib_VideoUpSample422_S16(mlib_s16 *dst, const  
mlib_s16 *src, mlib_s32 n);`

**DESCRIPTION** | The `mlib_VideoUpSample422_S16()` function performs up sampling rate conversion used in JPEG File Interchange Format (JFIF).

**PARAMETERS** | The function takes the following arguments:

*dst* | Pointer to destination row. *dst* must be 8-byte aligned.

*src* | Pointer to source row. *src* must be 8-byte aligned.

*n* | Length of source rows. *n* must be greater than 1.

**RETURN VALUES** | The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

**ATTRIBUTES** | See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Evolving        |
| MT-Level            | MT-Safe         |

**SEE ALSO** | `mlib_VideoDownSample420(3MLIB)`, `mlib_VideoDownSample420_S16(3MLIB)`, `mlib_VideoDownSample422(3MLIB)`, `mlib_VideoDownSample422_S16(3MLIB)`, `mlib_VideoUpSample420(3MLIB)`, `mlib_VideoUpSample420_Nearest(3MLIB)`, `mlib_VideoUpSample420_Nearest_S16(3MLIB)`, `mlib_VideoUpSample420_S16(3MLIB)`, `mlib_VideoUpSample422(3MLIB)`, `mlib_VideoUpSample422_Nearest(3MLIB)`, `mlib_VideoUpSample422_Nearest_S16(3MLIB)`, `attributes(5)`

mllib\_VideoWaveletForwardTwoTenTrans(3MLIB)

| <b>NAME</b>          | mllib_VideoWaveletForwardTwoTenTrans,<br>mllib_VideoWaveletForwardTwoTenTrans_S16_U8,<br>mllib_VideoWaveletForwardTwoTenTrans_S16_S16,<br>mllib_VideoWaveletForwardTwoTenTrans_S32_S16,<br>mllib_VideoWaveletForwardTwoTenTrans_S32_S32 – wavelet transformation                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |                |                 |                     |          |          |         |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|---------------------|----------|----------|---------|
| <b>SYNOPSIS</b>      | <pre>cc [ flag... ] file... -lmllib [ library... ] #include &lt;mllib.h&gt;  mllib_status mllib_VideoWaveletForwardTwoTenTrans_S16_U8(mllib_s16     *dst, const mllib_u8 *src, mllib_s32 width, mllib_s32 height,     mllib_s32 *level);  mllib_status mllib_VideoWaveletForwardTwoTenTrans_S16_S16(mllib_s16     *dst, const mllib_s16 *src, mllib_s32 width, mllib_s32 height,     mllib_s32 *level);  mllib_status mllib_VideoWaveletForwardTwoTenTrans_S32_S16(mllib_s32     *dst, const mllib_s16 *src, mllib_s32 width, mllib_s32 height,     mllib_s32 *level);  mllib_status mllib_VideoWaveletForwardTwoTenTrans_S32_S32(mllib_s32     *dst, const mllib_s32 *src, mllib_s32 width, mllib_s32 height,     mllib_s32 *level);</pre> |                |                 |                     |          |          |         |
| <b>DESCRIPTION</b>   | Each of the functions provides support to reversible wavelet transformation. It is for a forward two-ten transformation.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                |                 |                     |          |          |         |
| <b>PARAMETERS</b>    | <p>Each of the functions takes the following arguments:</p> <p><i>dst</i>                    Pointer to TT-transform coefficients.</p> <p><i>src</i>                    Pointer to source image.</p> <p><i>width</i>                 Width of image.</p> <p><i>height</i>                Height of image.</p> <p><i>level</i>                 Pointer to the number of decomposition levels. It returns the processed decomposition levels value.</p>                                                                                                                                                                                                                                                                                       |                |                 |                     |          |          |         |
| <b>RETURN VALUES</b> | Each of the functions returns MLIB_SUCCESS if successful. Otherwise it returns MLIB_FAILURE.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                |                 |                     |          |          |         |
| <b>ATTRIBUTES</b>    | See attributes(5) for descriptions of the following attributes:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                |                 |                     |          |          |         |
|                      | <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center;">ATTRIBUTE TYPE</th> <th style="text-align: center;">ATTRIBUTE VALUE</th> </tr> </thead> <tbody> <tr> <td>Interface Stability</td> <td>Evolving</td> </tr> <tr> <td>MT-Level</td> <td>MT-Safe</td> </tr> </tbody> </table>                                                                                                                                                                                                                                                                                                                                                                                                   | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Evolving | MT-Level | MT-Safe |
| ATTRIBUTE TYPE       | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                |                 |                     |          |          |         |
| Interface Stability  | Evolving                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                |                 |                     |          |          |         |
| MT-Level             | MT-Safe                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                |                 |                     |          |          |         |
| <b>SEE ALSO</b>      | <a href="#">mllib_VideoWaveletInverseTwoTenTrans(3MLIB)</a> , <a href="#">attributes(5)</a>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |                |                 |                     |          |          |         |



mllib\_VideoWaveletInverseTwoTenTrans(3MLIB)

**NAME** mllib\_VideoWaveletInverseTwoTenTrans,  
 mllib\_VideoWaveletInverseTwoTenTrans\_U8\_S16,  
 mllib\_VideoWaveletInverseTwoTenTrans\_S16\_S16,  
 mllib\_VideoWaveletInverseTwoTenTrans\_S16\_S32,  
 mllib\_VideoWaveletInverseTwoTenTrans\_S32\_S32 – wavelet transformation

**SYNOPSIS**

```
cc [flag...] file... -lmllib [library...]
#include <mllib.h>

mllib_status mllib_VideoWaveletInverseTwoTenTrans_U8_S16(mllib_u8
 *dst, const mllib_s16 *src, mllib_s32 width, mllib_s32 height,
 mllib_s32 *level);

mllib_status mllib_VideoWaveletInverseTwoTenTrans_S16_S16(mllib_s16
 *dst, const mllib_s16 *src, mllib_s32 width, mllib_s32 height,
 mllib_s32 *level);

mllib_status mllib_VideoWaveletInverseTwoTenTrans_S16_S32(mllib_s16
 *dst, const mllib_s32 *src, mllib_s32 width, mllib_s32 height,
 mllib_s32 *level);

mllib_status mllib_VideoWaveletInverseTwoTenTrans_S32_S32(mllib_s32
 *dst, const mllib_s32 *src, mllib_s32 width, mllib_s32 height,
 mllib_s32 *level);
```

**DESCRIPTION** Each of the functions provides support to reversible wavelet transformation. It is for an inverse two-ten transformation.

**PARAMETERS** Each of the functions takes the following arguments:

*dst* Pointer to destination image.

*src* Pointer to TT-transform coefficients.

*width* Width of image.

*height* Height of image.

*level* Pointer to the number of decomposition levels. It returns the processed decomposition levels value.

**RETURN VALUES** Each of the functions returns MLIB\_SUCCESS if successful. Otherwise it returns MLIB\_FAILURE.

**ATTRIBUTES** See attributes(5) for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Evolving        |
| MT-Level            | MT-Safe         |

**SEE ALSO** mllib\_VideoWaveletForwardTwoTenTrans(3MLIB), attributes(5)

mllib\_VolumeFindMaxBMask\_U8(3MLIB)

**NAME** mllib\_VolumeFindMaxBMask\_U8, mllib\_VolumeFindMaxBMask\_S16 – maximum intensity searching

**SYNOPSIS**

```
cc [flag...] file... -lmllib [library...]
#include <mllib.h>

mllib_status mllib_VolumeFindMaxBMask_U8(mllib_u8 *max, const
 mllib_rays *rays, const mllib_u8 *bmask);

mllib_status mllib_VolumeFindMaxBMask_S16(mllib_s16 *max, const
 mllib_rays *rays, const mllib_u8 *bmask);
```

**DESCRIPTION** Each function performs maximum intensity searching.

It uses the following equation:

$$\max[i] = \text{MAX}\{ \text{rays->results}[j][i] \mid j = 0, 1, \dots, \text{rays->nsteps}[i]; \text{bmask}[j] = 1 \}$$

where  $i = 0, 1, \dots, \text{rays->nrays} - 1$ .

**PARAMETERS** The function takes the following arguments:

*max* Pointer to an array of *rays->nrays* maximum values of the samples in each ray.

*rays* Pointer to an *mllib\_rays* structure. The data *rays->results* are organized with ray number (rather than ray step) varying fastest. Ray number and ray step are the output of the ray casting functions. The data might have values beyond the maximum step on a ray. For example, *rays->results[rays->nsteps[i]][i]* on ray *i* might not equal 0.

*bmask* Pointer to a 1-bit mask array. Eight mask bits are packed into one byte. A 1 corresponds to the data in the step to be considered.

**RETURN VALUES** The function returns *MLIB\_SUCCESS* if successful. Otherwise it returns *MLIB\_FAILURE*.

**ATTRIBUTES** See *attributes(5)* for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Evolving        |
| MT-Level            | MT-Safe         |

**SEE ALSO** *mllib\_VolumeFindMax\_U8(3MLIB)*, *mllib\_VolumeFindMaxCMask\_U8(3MLIB)*, *attributes(5)*

mllib\_VolumeFindMaxCMask\_U8(3MLIB)

**NAME** mllib\_VolumeFindMaxCMask\_U8, mllib\_VolumeFindMaxCMask\_S16 – maximum intensity searching

**SYNOPSIS**

```
cc [flag...] file... -lmllib [library...]
#include <mllib.h>

mllib_status mllib_VolumeFindMaxCMask_U8(mllib_u8 *max, const
 mllib_rays *rays, const mllib_u8 *cmask, mllib_s32 thresh);

mllib_status mllib_VolumeFindMaxCMask_S16(mllib_s16 *max, const
 mllib_rays *rays, const mllib_u8 *cmask, mllib_s32 thresh);
```

**DESCRIPTION** Each function performs maximum intensity searching.

It uses the following equation:

$$\max[i] = \text{MAX}\{ \text{rays} \rightarrow \text{results}[j][i] \mid j = 0, 1, \dots, \text{rays} \rightarrow \text{nsteps}[i]; \text{cmask}[j] > \text{thresh} \}$$

where  $i = 0, 1, \dots, \text{rays} \rightarrow \text{nrays} - 1$ .

**PARAMETERS** The function takes the following arguments:

*max* Pointer to an array of `rays->nrays` maximum values of the samples in each ray.

*rays* Pointer to an `mllib_rays` structure. The data `rays->results` are organized with ray number (rather than ray step) varying fastest. Ray number and ray step are the output of the ray casting functions. The data might have values beyond the maximum step on a ray. For example, `rays->results[rays->nsteps[i]][i]` on ray  $i$  might not equal 0.

*cmask* Pointer to an unsigned 8-bit mask array. If `cmask[j] > thresh`, then data in step  $j$ , `rays->results[j]`, are considered.

*thresh* Threshold.

**RETURN VALUES** The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

**ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Evolving        |
| MT-Level            | MT-Safe         |

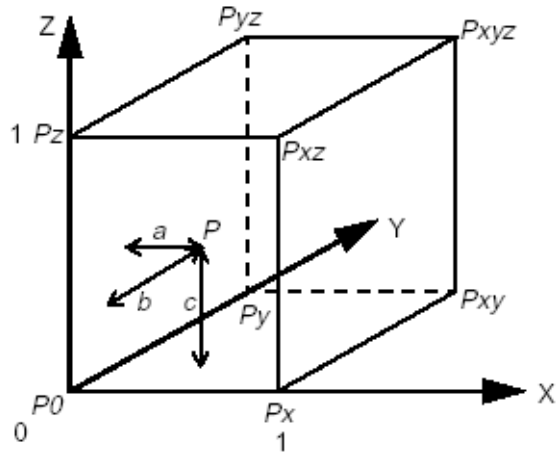
**SEE ALSO** `mllib_VolumeFindMax_U8(3MLIB)`, `mllib_VolumeFindMaxBMask_U8(3MLIB)`, `attributes(5)`

## mllib\_VolumeFindMax\_U8(3MLIB)

| <b>NAME</b>          | mllib_VolumeFindMax_U8, mllib_VolumeFindMax_S16 – maximum intensity searching                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |                |                 |                     |          |          |         |
|----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-----------------|---------------------|----------|----------|---------|
| <b>SYNOPSIS</b>      | <pre>cc [ flag... ] file... -lmllib [ library... ] #include &lt;mllib.h&gt;  mllib_status mllib_VolumeFindMax_U8(mllib_u8 *max, const mllib_rays     *rays);  mllib_status mllib_VolumeFindMax_S16(mllib_s16 *max, const mllib_rays     *rays);</pre>                                                                                                                                                                                                                                                                                                                                                        |                |                 |                     |          |          |         |
| <b>DESCRIPTION</b>   | <p>Each function performs maximum intensity searching.</p> <p>It uses the following equation:</p> $\max[i] = \text{MAX}\{ \text{rays->results}[j][i] \mid j = 0, 1, \dots, \text{rays->nsteps}[i] \}$ <p>where <math>i = 0, 1, \dots, \text{rays-&gt;nrays} - 1</math>.</p>                                                                                                                                                                                                                                                                                                                                  |                |                 |                     |          |          |         |
| <b>PARAMETERS</b>    | <p>The function takes the following arguments:</p> <p><i>max</i>                      Pointer to an array of <i>rays-&gt;nrays</i> maximum values of the samples in each ray.</p> <p><i>rays</i>                      Pointer to an <i>mllib_rays</i> structure. The data <i>rays-&gt;results</i> are organized with ray number (rather than ray step) varying fastest. Ray number and ray step are the output of the ray casting functions. The data might have values beyond the maximum step on a ray. For example, <i>rays-&gt;results[rays-&gt;nsteps[i]][i]</i> on ray <i>i</i> might not equal 0.</p> |                |                 |                     |          |          |         |
| <b>RETURN VALUES</b> | The function returns <i>MLIB_SUCCESS</i> if successful. Otherwise it returns <i>MLIB_FAILURE</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |                |                 |                     |          |          |         |
| <b>ATTRIBUTES</b>    | See <i>attributes(5)</i> for descriptions of the following attributes:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |                |                 |                     |          |          |         |
|                      | <table border="1"><thead><tr><th>ATTRIBUTE TYPE</th><th>ATTRIBUTE VALUE</th></tr></thead><tbody><tr><td>Interface Stability</td><td>Evolving</td></tr><tr><td>MT-Level</td><td>MT-Safe</td></tr></tbody></table>                                                                                                                                                                                                                                                                                                                                                                                             | ATTRIBUTE TYPE | ATTRIBUTE VALUE | Interface Stability | Evolving | MT-Level | MT-Safe |
| ATTRIBUTE TYPE       | ATTRIBUTE VALUE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                |                 |                     |          |          |         |
| Interface Stability  | Evolving                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                |                 |                     |          |          |         |
| MT-Level             | MT-Safe                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                |                 |                     |          |          |         |
| <b>SEE ALSO</b>      | <i>mllib_VolumeFindMaxBMask_U8(3MLIB)</i> ,<br><i>mllib_VolumeFindMaxCMask_U8(3MLIB)</i> , <i>attributes(5)</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |                |                 |                     |          |          |         |

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>NAME</b>        | <p>mllib_VolumeRayCast_Blocked,<br/> mllib_VolumeRayCast_Blocked_Parallel_Nearest_U8_U8,<br/> mllib_VolumeRayCast_Blocked_Parallel_Nearest_S16_S16,<br/> mllib_VolumeRayCast_Blocked_Parallel_Trilinear_U8_U8,<br/> mllib_VolumeRayCast_Blocked_Parallel_Trilinear_S16_S16,<br/> mllib_VolumeRayCast_Blocked_Divergent_Nearest_U8_U8,<br/> mllib_VolumeRayCast_Blocked_Divergent_Nearest_S16_S16,<br/> mllib_VolumeRayCast_Blocked_Divergent_Trilinear_U8_U8,<br/> mllib_VolumeRayCast_Blocked_Divergent_Trilinear_S16_S16 – cast a ray (or rays) through a 3D data set</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>SYNOPSIS</b>    | <pre>cc [ flag... ] file... -lmllib [ library... ] #include &lt;mllib.h&gt;  mllib_status mllib_VolumeRayCast_Blocked_Parallel_Nearest_U8_U8     (mllib_rays *rays, const mllib_blkvolume *blk, void *buffer);  mllib_status mllib_VolumeRayCast_Blocked_Parallel_Nearest_S16_S16     (mllib_rays *rays, const mllib_blkvolume *blk, void *buffer);  mllib_status mllib_VolumeRayCast_Blocked_Parallel_Trilinear_U8_U8     (mllib_rays *rays, const mllib_blkvolume *blk, void *buffer);  mllib_status     mllib_VolumeRayCast_Blocked_Parallel_Trilinear_S16_S16     (mllib_rays *rays, const mllib_blkvolume *blk, void *buffer);  mllib_status mllib_VolumeRayCast_Blocked_Divergent_Nearest_U8_U8     (mllib_rays *rays, const mllib_blkvolume *blk, void *buffer);  mllib_status mllib_VolumeRayCast_Blocked_Divergent_Nearest_S16_S16     (mllib_rays *rays, const mllib_blkvolume *blk, void *buffer);  mllib_status mllib_VolumeRayCast_Blocked_Divergent_Trilinear_U8_U8     (mllib_rays *rays, const mllib_blkvolume *blk, void *buffer);  mllib_status     mllib_VolumeRayCast_Blocked_Divergent_Trilinear_S16_S16     (mllib_rays *rays, const mllib_blkvolume *blk, void *buffer);</pre> |
| <b>DESCRIPTION</b> | <p>Each of these functions casts a ray (or rays) through a three-dimensional (3D) data set, then computes and returns the interpolated samples at each step along the way.</p> <p>In trilinear interpolation, the value at point P is computed from its eight surrounding neighbors based on the equation below.</p> $P = (1-a) * (1-b) * (1-c) * P0 +$ $a * (1-b) * (1-c) * Px + (1-a) * b * (1-c) * Py + (1-a) * (1-b) * c * Pz +$ $a * b * (1-c) * Pxy + a * (1-b) * c * Pxz + (1-a) * b * c * Pyz +$ $a * b * c * Pxyz$ <p>where a, b, and c are the fractional parts of the coordinates of point P.</p> <p>The trilinear interpolation is represented by the following figure:</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

mllib\_VolumeRayCast\_Blocked(3MLIB)



In nearest neighbor operation, the sample value at point P is replaced by the value of the nearest neighbor voxel.

**PARAMETERS** Each of the functions takes the following arguments:

- rays* Casting rays.
- blk* Volume data in the blocked format.
- buffer* Working buffer.

**RETURN VALUES** Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

**ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Evolving        |
| MT-Level            | MT-Safe         |

**SEE ALSO** [mllib\\_VolumeRayCast\\_General\(3MLIB\)](#), `attributes(5)`

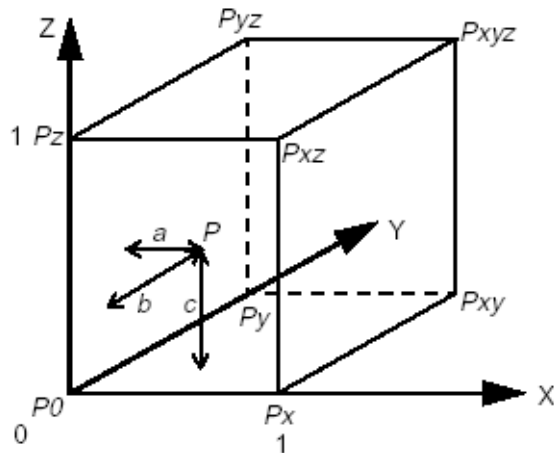
|                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>NAME</b>        | <p>mllib_VolumeRayCast_General,<br/> mllib_VolumeRayCast_General_Parallel_Nearest_U8_Bit,<br/> mllib_VolumeRayCast_General_Parallel_Nearest_U8_U8,<br/> mllib_VolumeRayCast_General_Parallel_Nearest_S16_S16,<br/> mllib_VolumeRayCast_General_Parallel_Trilinear_U8_U8,<br/> mllib_VolumeRayCast_General_Parallel_Trilinear_S16_S16,<br/> mllib_VolumeRayCast_General_Divergent_Nearest_U8_Bit,<br/> mllib_VolumeRayCast_General_Divergent_Nearest_U8_U8,<br/> mllib_VolumeRayCast_General_Divergent_Nearest_S16_S16,<br/> mllib_VolumeRayCast_General_Divergent_Trilinear_U8_U8,<br/> mllib_VolumeRayCast_General_Divergent_Trilinear_S16_S16 – cast a ray (or rays) through a 3D data set</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <b>SYNOPSIS</b>    | <pre>cc [ flag... ] file... -lmllib [ library... ] #include &lt;mllib.h&gt;  mllib_status mllib_VolumeRayCast_General_Parallel_Nearest_U8_Bit     (mllib_rays *rays, const mllib_genvolume *vol, void *buffer);  mllib_status mllib_VolumeRayCast_General_Parallel_Nearest_U8_U8     (mllib_rays *rays, const mllib_genvolume *vol, void *buffer);  mllib_status mllib_VolumeRayCast_General_Parallel_Nearest_S16_S16     (mllib_rays *rays, const mllib_genvolume *vol, void *buffer);  mllib_status mllib_VolumeRayCast_General_Parallel_Trilinear_U8_U8     (mllib_rays *rays, const mllib_genvolume *vol, void *buffer);  mllib_status     mllib_VolumeRayCast_General_Parallel_Trilinear_S16_S16     (mllib_rays *rays, const mllib_genvolume *vol, void *buffer);  mllib_status mllib_VolumeRayCast_General_Divergent_Nearest_U8_Bit     (mllib_rays *rays, const mllib_genvolume *vol, void *buffer);  mllib_status mllib_VolumeRayCast_General_Divergent_Nearest_U8_U8     (mllib_rays *rays, const mllib_genvolume *vol, void *buffer);  mllib_status mllib_VolumeRayCast_General_Divergent_Nearest_S16_S16     (mllib_rays *rays, const mllib_genvolume *vol, void *buffer);  mllib_status mllib_VolumeRayCast_General_Divergent_Trilinear_U8_U8     (mllib_rays *rays, const mllib_genvolume *vol, void *buffer);  mllib_status     mllib_VolumeRayCast_General_Divergent_Trilinear_S16_S16     (mllib_rays *rays, const mllib_genvolume *vol, void *buffer);</pre> |
| <b>DESCRIPTION</b> | <p>Each of these functions casts a ray (or rays) through a three-dimensional (3D) data set, then computes and returns the interpolated samples at each step along the way.</p> <p>In trilinear interpolation, the value at point P is computed from its eight surrounding neighbors based on the equation below.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

## mllib\_VolumeRayCast\_General(3MLIB)

$$P = (1-a)*(1-b)*(1-c)*P_0 + a*(1-b)*(1-c)*P_x + (1-a)*b*(1-c)*P_y + (1-a)*(1-b)*c*P_z + a*b*(1-c)*P_{xy} + a*(1-b)*c*P_{xz} + (1-a)*b*c*P_{yz} + a*b*c*P_{xyz}$$

where  $a$ ,  $b$ , and  $c$  are the fractional parts of the coordinates of point  $P$ .

The trilinear interpolation is represented by the following figure:



In nearest neighbor operation, the sample value at point  $P$  is replaced by the value of the nearest neighbor voxel.

**PARAMETERS** Each of the functions takes the following arguments:

*rays* Casting rays.  
*vol* Volume data that consists of slices.  
*buffer* Working buffer.

**RETURN VALUES** Each of the functions returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

**ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Evolving        |
| MT-Level            | MT-Safe         |

**SEE ALSO** [mllib\\_VolumeRayCast\\_Blocked\(3MLIB\)](#), `attributes(5)`



**NAME** mllib\_VolumeWindowLevel – window-level operation

**SYNOPSIS** `cc [ flag... ] file... -lmllib [ library... ]`  
`#include <mllib.h>`

```
mllib_status mllib_VolumeWindowLevel(mllib_u8 *dst, const mllib_s16
 *src, mllib_s32 window, mllib_s32 level, mllib_s32 gmax, mllib_s32
 gmin, mllib_s32 len);
```

**DESCRIPTION** The `mllib_VolumeWindowLevel()` function performs a window-level operation by using the following equation:

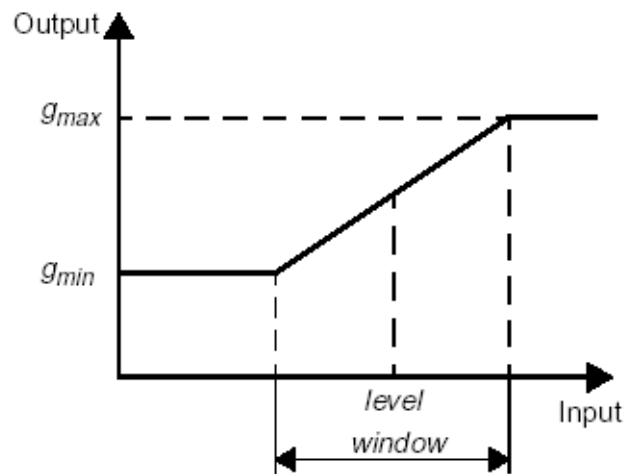
$$X = x(i) \quad i = 0, 1, \dots, n-1$$

$$Z = z(i) \quad i = 0, 1, \dots, n-1$$

$$= \begin{cases} g_{min} & x(i) < l - \frac{W}{2} \\ (g_{max} - g_{min}) \frac{x(i) - (l - \frac{W}{2})}{W} + g_{min} & l - \frac{W}{2} \leq x(i) < l + \frac{W}{2} \\ g_{max} & x(i) \geq l + \frac{W}{2} \end{cases}$$

The window-level operation is represented by the following figure:

mllib\_VolumeWindowLevel(3MLIB)



**PARAMETERS** The function takes the following arguments:

- dst* Pointer to the output or destination array.
- src* Pointer to the input or source array.
- window* Width of the window.
- level* Center of the window.
- gmax* Maximum grayscale in the destination array.
- gmin* Minimum grayscale in the destination array.
- len* Length of the data array.

**RETURN VALUES** The function returns `MLIB_SUCCESS` if successful. Otherwise it returns `MLIB_FAILURE`.

**ATTRIBUTES** See `attributes(5)` for descriptions of the following attributes:

| ATTRIBUTE TYPE      | ATTRIBUTE VALUE |
|---------------------|-----------------|
| Interface Stability | Evolving        |
| MT-Level            | MT-Safe         |

**SEE ALSO** `attributes(5)`

# Index

---

## Numbers and Symbols

- 0.5X zoom — `mlib_ImageZoomOut2X`, 770
- 0.5X zoom —
  - `mlib_ImageZoomOut2XIndex`, 773
- 0.5X zoom — `mlib_ImageZoomOut2X_Fp`, 771
- 16-bit to 8-bit quantization —
  - `mlib_SignalQuant_U8S_S16S`, 1163
- 16-bit to 8-bit quantization —
  - `mlib_SignalQuant_U8_S16`, 1163
- 2X zoom — `mlib_ImageZoomIn2X`, 763
- 2X zoom — `mlib_ImageZoomIn2X_Fp`, 764
- 2X zoom on color-indexed image —
  - `mlib_ImageZoomIn2XIndex`, 766
- 2x2 convolution — `mlib_ImageConv2x2`, 371
- 2x2 convolution —
  - `mlib_ImageConv2x2_Fp`, 373
- 2x2 convolution on a color indexed image —
  - `mlib_ImageConv2x2Index`, 375
- 3x3 convolution — `mlib_ImageConv3x3`, 377
- 3x3 convolution —
  - `mlib_ImageConv3x3_Fp`, 379
- 3x3 convolution on a color indexed image —
  - `mlib_ImageConv3x3Index`, 381
- 3x3 gradient filter —
  - `mlib_ImageGradient3x3`, 481
- 3x3 gradient filter —
  - `mlib_ImageGradient3x3_Fp`, 483
- 3x3 maximum filter —
  - `mlib_ImageMaxFilter3x3`, 536
- 3x3 maximum filter —
  - `mlib_ImageMaxFilter3x3_Fp`, 538
- 3x3 median filter —
  - `mlib_ImageMedianFilter3x3`, 555
- 3x3 median filter —
  - `mlib_ImageMedianFilter3x3_Fp`, 557
- 3x3 median filter, unsigned —
  - `mlib_ImageMedianFilter3x3_US`, 559
- 3x3 Min Filter — `mlib_ImageMinFilter3x3`, 580
- 3x3 Min Filter, floating point —
  - `mlib_ImageMinFilter3x3_Fp`, 582
- 3x3 rank filter —
  - `mlib_ImageRankFilter3x3`, 632
- 3x3 rank filter —
  - `mlib_ImageRankFilter3x3_Fp`, 634
- 3x3 rank filter, unsigned —
  - `mlib_ImageRankFilter3x3_US`, 636
- 4x4 convolution — `mlib_ImageConv4x4`, 383
- 4x4 convolution —
  - `mlib_ImageConv4x4_Fp`, 385
- 4x4 convolution on a color indexed image —
  - `mlib_ImageConv4x4Index`, 387
- 5x5 convolution — `mlib_ImageConv5x5`, 389
- 5x5 convolution —
  - `mlib_ImageConv5x5_Fp`, 391
- 5x5 convolution on a color indexed image —
  - `mlib_ImageConv5x5Index`, 393
- 5x5 Max Filter — `mlib_ImageMaxFilter5x5`, 540
- 5x5 Max Filter —
  - `mlib_ImageMaxFilter5x5_Fp`, 542
- 5x5 median filter —
  - `mlib_ImageMedianFilter5x5`, 561
- 5x5 median filter —
  - `mlib_ImageMedianFilter5x5_Fp`, 563
- 5x5 median filter, unsigned —
  - `mlib_ImageMedianFilter5x5_US`, 565
- 5x5 Min Filter — `mlib_ImageMinFilter5x5`, 584

- 5x5 Min Filter, floating point —  
  - `mllib_ImageMinFilter5x5_Fp`, 586
- 5x5 rank filter —  
  - `mllib_ImageRankFilter5x5`, 638
- 5x5 rank filter —  
  - `mllib_ImageRankFilter5x5_Fp`, 640
- 5x5 rank filter, unsigned —  
  - `mllib_ImageRankFilter5x5_US`, 642
- 7x7 convolution — `mllib_ImageConv7x7`, 395
- 7x7 convolution —  
  - `mllib_ImageConv7x7_Fp`, 397
- 7x7 convolution on a color indexed image —  
  - `mllib_ImageConv7x7Index`, 399
- 7x7 Max Filter — `mllib_ImageMaxFilter7x7`, 544
- 7x7 Max Filter —  
  - `mllib_ImageMaxFilter7x7_Fp`, 546
- 7x7 median filter —  
  - `mllib_ImageMedianFilter7x7`, 567
- 7x7 median filter —  
  - `mllib_ImageMedianFilter7x7_Fp`, 569
- 7x7 median filter, unsigned —  
  - `mllib_ImageMedianFilter7x7_US`, 571
- 7x7 Min Filter — `mllib_ImageMinFilter7x7`, 588
- 7x7 Min Filter, floating point —  
  - `mllib_ImageMinFilter7x7_Fp`, 590
- 7x7 rank filter —  
  - `mllib_ImageRankFilter7x7`, 644
- 7x7 rank filter —  
  - `mllib_ImageRankFilter7x7_Fp`, 646
- 7x7 rank filter, unsigned —  
  - `mllib_ImageRankFilter7x7_US`, 648

## A

- ABGR to JFIF YCbCr color conversion —  
  - `mllib_VideoColorABGR2JFIFYCC420`, 1291
- ABGR to JFIF YCbCr color conversion —  
  - `mllib_VideoColorABGR2JFIFYCC422`, 1292
- ABGR to JFIF YCbCr color conversion —  
  - `mllib_VideoColorABGR2JFIFYCC444`, 1293
- adaptive differential pulse code modulation (ADPCM) —  
  - `mllib_SignalADPCM2Bits2Linear`, 849
- adaptive differential pulse code modulation (ADPCM) —  
  - `mllib_SignalADPCM3Bits2Linear`, 850
- adaptive differential pulse code modulation (ADPCM) —  
  - `mllib_SignalADPCM4Bits2Linear`, 851
- adaptive differential pulse code modulation (ADPCM) —  
  - `mllib_SignalADPCM5Bits2Linear`, 852
- adaptive differential pulse code modulation (ADPCM) — `mllib_SignalADPCMFree`, 853
- adaptive differential pulse code modulation (ADPCM) — `mllib_SignalADPCMInit`, 854
- adaptive differential pulse code modulation (ADPCM) —  
  - `mllib_SignalLinear2ADPCM2Bits`, 1044
- adaptive differential pulse code modulation (ADPCM) —  
  - `mllib_SignalLinear2ADPCM3Bits`, 1045
- adaptive differential pulse code modulation (ADPCM) —  
  - `mllib_SignalLinear2ADPCM4Bits`, 1046
- adaptive differential pulse code modulation (ADPCM) —  
  - `mllib_SignalLinear2ADPCM5Bits`, 1047
- addition with a constant —  
  - `mllib_ImageConstAdd`, 335
- addition with a constant —  
  - `mllib_ImageConstAdd_Fp`, 336
- addition with a constant —  
  - `mllib_ImageConstAdd_Fp_Inp`, 337
- addition with a constant —  
  - `mllib_ImageConstAdd_Inp`, 338
- adds motion-compensated 8x8 block to the current block —  
  - `mllib_VideoAddBlock_U8_S16`, 1289
- affine transformation on a color indexed image — `mllib_ImageAffineIndex`, 235
- affine transformation on a color indexed image, checking the matrix first —  
  - `mllib_ImageAffineTransformIndex`, 245
- affine transformation on an image, checking the matrix first —  
  - `mllib_ImageAffineTransform`, 241
- affine transformation on an image, checking the matrix first —  
  - `mllib_ImageAffineTransform_Fp`, 243
- affine transformation on an image with table-driven interpolation —  
  - `mllib_ImageAffineTable`, 237

- affine transformation on an image with table-driven interpolation —
  - mllib\_ImageAffineTable\_Fp, 239
- allocate a block of bytes — mllib\_malloc, 792
- alpha channel division —
  - mllib\_ImageDivAlpha, 433
- alpha channel division —
  - mllib\_ImageDivAlpha\_Fp, 435
- alpha channel division, in place —
  - mllib\_ImageDivAlpha\_Fp\_Inp, 436
- alpha channel division, in place —
  - mllib\_ImageDivAlpha\_Inp, 437
- alpha channel multiplication —
  - mllib\_ImageMulAlpha, 599
- alpha channel multiplication —
  - mllib\_ImageMulAlpha\_Fp, 600
- alpha channel multiplication —
  - mllib\_ImageMulAlpha\_Fp\_Inp, 601
- alpha channel multiplication, in place —
  - mllib\_ImageMulAlpha\_Inp, 602
- And with a constant —
  - mllib\_ImageConstAnd, 339
- And with a constant —
  - mllib\_ImageConstAnd\_Inp, 340
- AndNot with a constant —
  - mllib\_ImageConstAndNot, 341
- AndNot with a constant —
  - mllib\_ImageConstAndNot\_Inp, 342
- anti-diagonal flip —
  - mllib\_ImageFlipAntiDiag, 461
- anti-diagonal flip —
  - mllib\_ImageFlipAntiDiag\_Fp, 462
- antialias filters and subsamples an image —
  - mllib\_ImageFilteredSubsample, 458
- antialias filters and subsamples an image —
  - mllib\_ImageFilteredSubsample\_Fp, 458
- applies a 2-dimensional (2D) 3x3 spatial filter on the reference block —
  - mllib\_VideoP64Loop\_S16\_U8, 1513
- applies a 2-dimensional (2D) 3x3 spatial filter on the reference block —
  - mllib\_VideoP64Loop\_U8\_U8, 1515
- ARGB to JFIF YCbCr color conversion —
  - mllib\_VideoColorARGB2JFIFYCC420, 1296
- ARGB to JFIF YCbCr color conversion —
  - mllib\_VideoColorARGB2JFIFYCC422, 1297
- ARGB to JFIF YCbCr color conversion —
  - mllib\_VideoColorARGB2JFIFYCC444, 1298

- auto-correlation of an image —
  - mllib\_ImageAutoCorrel, 252
- auto-correlation of an image —
  - mllib\_ImageAutoCorrel\_Fp, 253
- average of two images — mllib\_ImageAve, 254
- average of two images —
  - mllib\_ImageAve\_Fp, 255
- average of two images, in place —
  - mllib\_ImageAve\_Fp\_Inp, 256
- average of two images, in place —
  - mllib\_ImageAve\_Inp, 257
- averages the source raster image over 2x2 blocks and writes the results to the destination raster image —
  - mllib\_VideoP64Decimate\_U8\_U8, 1511

## B

- Bartlett windowing multiplication —
  - mllib\_SignalMulBartlett\_F32, 1110
- Bartlett windowing multiplication —
  - mllib\_SignalMulBartlett\_F32S, 1110
- Bartlett windowing multiplication —
  - mllib\_SignalMulBartlett\_F32S\_F32S, 1111
- Bartlett windowing multiplication —
  - mllib\_SignalMulBartlett\_F32\_F32, 1111
- Bartlett windowing multiplication —
  - mllib\_SignalMulBartlett\_S16, 1112
- Bartlett windowing multiplication —
  - mllib\_SignalMulBartlett\_S16S, 1112
- Bartlett windowing multiplication —
  - mllib\_SignalMulBartlett\_S16S\_S16S, 1113
- Bartlett windowing multiplication —
  - mllib\_SignalMulBartlett\_S16\_S16, 1113
- Bartlett windowing multiplication —
  - mllib\_SignalMulHamming\_F32, 1120
- Bartlett windowing multiplication —
  - mllib\_SignalMulHamming\_F32S, 1120
- Bartlett windowing multiplication —
  - mllib\_SignalMulHamming\_S16, 1122
- Bartlett windowing multiplication —
  - mllib\_SignalMulHamming\_S16S, 1122
- Bartlett window generation —
  - mllib\_SignalGenBartlett\_F32, 992
- Bartlett window generation —
  - mllib\_SignalGenBartlett\_S16, 993

- biquad Infinite Impulse Response (IIR) filtering
  - `mllib_SignalIIRFree_Biquad_F32S_F32S`, 1027
- biquad Infinite Impulse Response (IIR) filtering
  - `mllib_SignalIIRFree_Biquad_F32_F32`, 1027
- biquad Infinite Impulse Response (IIR) filtering
  - `mllib_SignalIIRFree_Biquad_S16S_S16S`, 1028
- biquad Infinite Impulse Response (IIR) filtering
  - `mllib_SignalIIRFree_Biquad_S16_S16`, 1028
- biquad Infinite Impulse Response (IIR) filtering
  - `mllib_SignalIIRInit_Biquad_F32S_F32S`, 1031
- biquad Infinite Impulse Response (IIR) filtering
  - `mllib_SignalIIRInit_Biquad_F32_F32`, 1031
- biquad Infinite Impulse Response (IIR) filtering
  - `mllib_SignalIIRInit_Biquad_S16S_S16S`, 1032
- biquad Infinite Impulse Response (IIR) filtering
  - `mllib_SignalIIRInit_Biquad_S16_S16`, 1032
- biquad Infinite Impulse Response (IIR) filtering
  - `mllib_SignalIIR_Biquad_F32S_F32S`, 1025
- biquad Infinite Impulse Response (IIR) filtering
  - `mllib_SignalIIR_Biquad_F32_F32`, 1025
- biquad Infinite Impulse Response (IIR) filtering
  - `mllib_SignalIIR_Biquad_S16S_S16S_Sat`, 1025
- biquad Infinite Impulse Response (IIR) filtering
  - `mllib_SignalIIR_Biquad_S16_S16_Sat`, 1025
- Blackman windowing multiplication —
  - `mllib_SignalMulBlackman_F32`, 1114
- Blackman windowing multiplication —
  - `mllib_SignalMulBlackman_F32S`, 1114
- Blackman windowing multiplication —
  - `mllib_SignalMulBlackman_F32S_F32S`, 1115
- Blackman windowing multiplication —
  - `mllib_SignalMulBlackman_F32_F32`, 1115
- Blackman windowing multiplication —
  - `mllib_SignalMulBlackman_S16`, 1116
- Blackman windowing multiplication —
  - `mllib_SignalMulBlackman_S16S`, 1116
- Blackman windowing multiplication —
  - `mllib_SignalMulBlackman_S16S_S16S`, 1117
- Blackman windowing multiplication —
  - `mllib_SignalMulBlackman_S16_S16`, 1117
- Blackman window generation —
  - `mllib_SignalGenBlackman_F32`, 994
- Blackman window generation —
  - `mllib_SignalGenBlackman_S16`, 995
- blend an image and a color —
  - `mllib_ImageBlendColor`, 269
- blend an image and a color —
  - `mllib_ImageBlendColor_Fp`, 270
- blend an image and a color, in place —
  - `mllib_ImageBlendColor_Fp_Inp`, 271
- blend an image and a color, in place —
  - `mllib_ImageBlendColor_Inp`, 272
- blend multiple images —
  - `mllib_ImageBlendMulti`, 274
- blend multiple images —
  - `mllib_ImageBlendMulti_Fp`, 276
- blend with an alpha image —
  - `mllib_ImageBlend1_Fp_Inp`, 258
- blend with an alpha image —
  - `mllib_ImageBlend2_Fp_Inp`, 260
- blend with an alpha image —
  - `mllib_ImageBlend`, 262
- blend with an alpha image —
  - `mllib_ImageBlend_Fp`, 273
- blend with an alpha image, in place —
  - `mllib_ImageBlend1_Inp`, 259
- blend with an alpha image, in place —
  - `mllib_ImageBlend2_Inp`, 261
- blending —
  - `mllib_ImageBlend_BSRC1_BSRC2`, 263
- blending — `mllib_ImageBlend_DA_DA`, 263
- blending — `mllib_ImageBlend_DA_DC`, 263
- blending —
  - `mllib_ImageBlend_DA_OMDA`, 263
- blending — `mllib_ImageBlend_DA_OMDC`, 263
- blending — `mllib_ImageBlend_DA_OMSA`, 263
- blending — `mllib_ImageBlend_DA_ONE`, 263
- blending — `mllib_ImageBlend_DA_SA`, 263
- blending — `mllib_ImageBlend_DA_SAS`, 263
- blending — `mllib_ImageBlend_DA_ZERO`, 263
- blending —
  - `mllib_ImageBlend_OMDA_DA`, 263
- blending — `mllib_ImageBlend_OMDA_DC`, 263
- blending —
  - `mllib_ImageBlend_OMDA_OMDA`, 263
- blending —
  - `mllib_ImageBlend_OMDA_OMDC`, 263

blending —  
     mlib\_ImageBlend\_OMDA\_OMSA, 263  
 blending —  
     mlib\_ImageBlend\_OMDA\_ONE, 263  
 blending — mlib\_ImageBlend\_OMDA\_SA, 263  
 blending —  
     mlib\_ImageBlend\_OMDA\_SAS, 263  
 blending —  
     mlib\_ImageBlend\_OMDA\_ZERO, 263  
 blending — mlib\_ImageBlend\_OMSA\_DA, 263  
 blending — mlib\_ImageBlend\_OMSA\_DC, 263  
 blending —  
     mlib\_ImageBlend\_OMSA\_OMDA, 263  
 blending —  
     mlib\_ImageBlend\_OMSA\_OMDC, 263  
 blending —  
     mlib\_ImageBlend\_OMSA\_OMSA, 263  
 blending —  
     mlib\_ImageBlend\_OMSA\_ONE, 263  
 blending — mlib\_ImageBlend\_OMSA\_SA, 263  
 blending —  
     mlib\_ImageBlend\_OMSA\_SAS, 263  
 blending —  
     mlib\_ImageBlend\_OMSA\_ZERO, 263  
 blending — mlib\_ImageBlend\_OMSC\_DA, 263  
 blending — mlib\_ImageBlend\_OMSC\_DC, 263  
 blending —  
     mlib\_ImageBlend\_OMSC\_OMDA, 263  
 blending —  
     mlib\_ImageBlend\_OMSC\_OMDC, 263  
 blending —  
     mlib\_ImageBlend\_OMSC\_OMSA, 263  
 blending —  
     mlib\_ImageBlend\_OMSC\_ONE, 263  
 blending — mlib\_ImageBlend\_OMSC\_SA, 263  
 blending —  
     mlib\_ImageBlend\_OMSC\_SAS, 263  
 blending —  
     mlib\_ImageBlend\_OMSC\_ZERO, 263  
 blending — mlib\_ImageBlend\_ONE\_DA, 263  
 blending — mlib\_ImageBlend\_ONE\_DC, 263  
 blending —  
     mlib\_ImageBlend\_ONE\_OMDA, 263  
 blending —  
     mlib\_ImageBlend\_ONE\_OMDC, 263  
 blending —  
     mlib\_ImageBlend\_ONE\_OMSA, 263  
 blending — mlib\_ImageBlend\_ONE\_ONE, 263  
 blending — mlib\_ImageBlend\_ONE\_SA, 263  
 blending — mlib\_ImageBlend\_ONE\_SAS, 263  
 blending —  
     mlib\_ImageBlend\_ONE\_ZERO, 263  
 blending — mlib\_ImageBlend\_SA\_DA, 263  
 blending — mlib\_ImageBlend\_SA\_DC, 263  
 blending — mlib\_ImageBlend\_SA\_OMDA, 263  
 blending — mlib\_ImageBlend\_SA\_OMDC, 263  
 blending — mlib\_ImageBlend\_SA\_OMSA, 263  
 blending — mlib\_ImageBlend\_SA\_ONE, 263  
 blending — mlib\_ImageBlend\_SA\_SA, 263  
 blending — mlib\_ImageBlend\_SA\_SAS, 263  
 blending — mlib\_ImageBlend\_SA\_ZERO, 263  
 blending — mlib\_ImageBlend\_SC\_DA, 263  
 blending — mlib\_ImageBlend\_SC\_DC, 263  
 blending — mlib\_ImageBlend\_SC\_OMDA, 263  
 blending — mlib\_ImageBlend\_SC\_OMDC, 263  
 blending — mlib\_ImageBlend\_SC\_OMSA, 263  
 blending — mlib\_ImageBlend\_SC\_ONE, 263  
 blending — mlib\_ImageBlend\_SC\_SA, 263  
 blending — mlib\_ImageBlend\_SC\_SAS, 263  
 blending — mlib\_ImageBlend\_SC\_ZERO, 263  
 blending — mlib\_ImageBlend\_ZERO\_DA, 263  
 blending — mlib\_ImageBlend\_ZERO\_DC, 263  
 blending —  
     mlib\_ImageBlend\_ZERO\_OMDA, 263  
 blending —  
     mlib\_ImageBlend\_ZERO\_OMDC, 263  
 blending —  
     mlib\_ImageBlend\_ZERO\_OMSA, 263  
 blending —  
     mlib\_ImageBlend\_ZERO\_ONE, 263  
 blending — mlib\_ImageBlend\_ZERO\_SA, 263  
 blending — mlib\_ImageBlend\_ZERO\_SAS, 263  
 blending —  
     mlib\_ImageBlend\_ZERO\_ZERO, 263  
 blending, in place —  
     mlib\_ImageBlend\_BSRC1\_BSRC2\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_DA\_DA\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_DA\_DC\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_DA\_OMDA\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_DA\_OMDC\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_DA\_OMSA\_Inp, 266

blending, in place —  
     mlib\_ImageBlend\_DA\_ONE\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_DA\_SAS\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_DA\_SA\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_DA\_ZERO\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMDA\_DA\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMDA\_DC\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMDA\_OMDA\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMDA\_OMDC\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMDA\_OMSA\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMDA\_ONE\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMDA\_SAS\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMDA\_SA\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMDA\_ZERO\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMSA\_DA\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMSA\_DC\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMSA\_OMDA\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMSA\_OMDC\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMSA\_OMSA\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMSA\_ONE\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMSA\_SAS\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMSA\_SA\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMSA\_ZERO\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMSC\_DA\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMSC\_DC\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMSC\_OMDA\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMSC\_OMDC\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMSC\_OMSA\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMSC\_ONE\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMSC\_SAS\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMSC\_SA\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_OMSC\_ZERO\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_ONE\_DA\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_ONE\_DC\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_ONE\_OMDA\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_ONE\_OMDC\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_ONE\_OMSA\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_ONE\_ONE\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_ONE\_SAS\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_ONE\_SA\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_ONE\_ZERO\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_SA\_DA\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_SA\_DC\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_SA\_OMDA\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_SA\_OMDC\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_SA\_OMSA\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_SA\_ONE\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_SA\_SAS\_Inp, 266  
 blending, in place —  
     mlib\_ImageBlend\_SA\_SA\_Inp, 266



- blending, in place —
  - mllib\_ImageBlend\_SA\_ZERO\_Inp, 266
- blending, in place —
  - mllib\_ImageBlend\_SC\_DA\_Inp, 266
- blending, in place —
  - mllib\_ImageBlend\_SC\_DC\_Inp, 266
- blending, in place —
  - mllib\_ImageBlend\_SC\_OMDA\_Inp, 266
- blending, in place —
  - mllib\_ImageBlend\_SC\_OMDC\_Inp, 266
- blending, in place —
  - mllib\_ImageBlend\_SC\_OMSA\_Inp, 266
- blending, in place —
  - mllib\_ImageBlend\_SC\_ONE\_Inp, 266
- blending, in place —
  - mllib\_ImageBlend\_SC\_SAS\_Inp, 266
- blending, in place —
  - mllib\_ImageBlend\_SC\_SA\_Inp, 266
- blending, in place —
  - mllib\_ImageBlend\_SC\_ZERO\_Inp, 266
- blending, in place —
  - mllib\_ImageBlend\_ZERO\_DA\_Inp, 266
- blending, in place —
  - mllib\_ImageBlend\_ZERO\_DC\_Inp, 266
- blending, in place —
  - mllib\_ImageBlend\_ZERO\_OMDA\_Inp, 266
- blending, in place —
  - mllib\_ImageBlend\_ZERO\_OMDC\_Inp, 266
- blending, in place —
  - mllib\_ImageBlend\_ZERO\_OMSA\_Inp, 266
- blending, in place —
  - mllib\_ImageBlend\_ZERO\_ONE\_Inp, 266
- blending, in place —
  - mllib\_ImageBlend\_ZERO\_SAS\_Inp, 266
- blending, in place —
  - mllib\_ImageBlend\_ZERO\_SA\_Inp, 266
- blending, in place —
  - mllib\_ImageBlend\_ZERO\_ZERO\_Inp, 266
- boundary fill —
  - mllib\_GraphicsBoundaryFill\_32, 41
- boundary fill —
  - mllib\_GraphicsBoundaryFill\_8, 41

## C

- cast a ray (or rays) through a 3D data set —
  - mllib\_VolumeRayCast\_Blocked, 1541

- cast a ray (or rays) through a 3D data set —
  - mllib\_VolumeRayCast\_Blocked\_Divergent\_Nearest\_S16\_S16, 1541
- cast a ray (or rays) through a 3D data set —
  - mllib\_VolumeRayCast\_Blocked\_Divergent\_Nearest\_U8\_U8, 1541
- cast a ray (or rays) through a 3D data set —
  - mllib\_VolumeRayCast\_Blocked\_Divergent\_Trilinear\_U8\_U8, 1541
- cast a ray (or rays) through a 3D data set —
  - mllib\_VolumeRayCast\_Blocked\_Divergent\_Trilinear\_S16\_S16, 1541
- cast a ray (or rays) through a 3D data set —
  - mllib\_VolumeRayCast\_Blocked\_Parallel\_Nearest\_S16\_S16, 1541
- cast a ray (or rays) through a 3D data set —
  - mllib\_VolumeRayCast\_Blocked\_Parallel\_Nearest\_U8\_U8, 1541
- cast a ray (or rays) through a 3D data set —
  - mllib\_VolumeRayCast\_Blocked\_Parallel\_Trilinear\_S16\_S16, 1541
- cast a ray (or rays) through a 3D data set —
  - mllib\_VolumeRayCast\_Blocked\_Parallel\_Trilinear\_U8\_U8, 1541
- cast a ray (or rays) through a 3D data set —
  - mllib\_VolumeRayCast\_General, 1543
- cast a ray (or rays) through a 3D data set —
  - mllib\_VolumeRayCast\_General\_Divergent\_Nearest\_S16\_S16, 1543
- cast a ray (or rays) through a 3D data set —
  - mllib\_VolumeRayCast\_General\_Divergent\_Nearest\_U8\_Bit, 1543
- cast a ray (or rays) through a 3D data set —
  - mllib\_VolumeRayCast\_General\_Divergent\_Nearest\_U8\_U8, 1543
- cast a ray (or rays) through a 3D data set —
  - mllib\_VolumeRayCast\_General\_Divergent\_Trilinear\_S16\_S16, 1543
- cast a ray (or rays) through a 3D data set —
  - mllib\_VolumeRayCast\_General\_Divergent\_Trilinear\_U8\_U8, 1543
- cast a ray (or rays) through a 3D data set —
  - mllib\_VolumeRayCast\_General\_Parallel\_Nearest\_S16\_S16, 1543
- cast a ray (or rays) through a 3D data set —
  - mllib\_VolumeRayCast\_General\_Parallel\_Nearest\_U8\_Bit, 1543

cast a ray (or rays) through a 3D data set —  
     mllib\_VolumeRayCast\_General\_Parallel\_Nearest\_U8\_U8, 1543  
 cast a ray (or rays) through a 3D data set —  
     mllib\_VolumeRayCast\_General\_Parallel\_Trilinear\_S16\_S16, 1543  
 cast a ray (or rays) through a 3D data set —  
     mllib\_VolumeRayCast\_General\_Parallel\_Trilinear\_U8\_U8, 1543  
 channel copy — mllib\_ImageChannelCopy, 280  
 channel extract —  
     mllib\_ImageChannelExtract, 281  
 channel insert —  
     mllib\_ImageChannelInsert, 282  
 channel merge —  
     mllib\_ImageChannelMerge, 283  
 channel split — mllib\_ImageChannelSplit, 284  
 clean up for autocorrelation method —  
     mllib\_SignalLPCAutoCorrelFree\_F32, 1068  
 clean up for autocorrelation method —  
     mllib\_SignalLPCAutoCorrelFree\_S16, 1068  
 clean up for cepstral analysis —  
     mllib\_SignalCepstralFree\_F32, 861  
 clean up for cepstral analysis —  
     mllib\_SignalCepstralFree\_S16, 861  
 clean up for cepstral analysis in mel frequency scale —  
     mllib\_SignalMelCepstralFree\_F32, 1101  
 clean up for cepstral analysis in mel frequency scale —  
     mllib\_SignalMelCepstralFree\_S16, 1101  
 clean up for covariance method —  
     mllib\_SignalLPCCovarianceFree\_F32, 1082  
 clean up for covariance method —  
     mllib\_SignalLPCCovarianceFree\_S16, 1082  
 clean up for K-best paths of scalar data —  
     mllib\_SignalDTWKScalarFree\_F32, 883  
 clean up for K-best paths of scalar data —  
     mllib\_SignalDTWKScalarFree\_S16, 883  
 clean up for K-best paths of vector data —  
     mllib\_SignalDTWKVectorFree\_F32, 898  
 clean up for K-best paths of vector data —  
     mllib\_SignalDTWKVectorFree\_S16, 898  
 clean up for perceptual weighting —  
     mllib\_SignalLPCPerceptWeightFree\_F32, 1087  
 clean up for perceptual weighting —  
     mllib\_SignalLPCPerceptWeightFree\_S16, 1087  
 clean up for scalar data —  
     mllib\_SignalDTWScalarFree\_F32, 915  
 clean up for scalar data —  
     mllib\_SignalDTWScalarFree\_S16, 915  
 clean up for signal pre-emphasizing —  
     mllib\_SignalEmphasizeFree\_F32S\_F32S, 950  
 clean up for signal pre-emphasizing —  
     mllib\_SignalEmphasizeFree\_F32\_F32, 950  
 clean up for signal pre-emphasizing —  
     mllib\_SignalEmphasizeFree\_S16S\_S16S, 950  
 clean up for signal pre-emphasizing —  
     mllib\_SignalEmphasizeFree\_S16\_S16, 950  
 clean up for vector data —  
     mllib\_SignalDTWVectorFree\_F32, 934  
 clean up for vector data —  
     mllib\_SignalDTWVectorFree\_S16, 934  
 clear — mllib\_ImageClear, 285  
 clear — mllib\_ImageClear\_Fp, 288  
 CMYK to JFIF YCbCr color conversion —  
     mllib\_VideoColorCMYK2JFIFCYCK444, 1307  
 color conversion —  
     mllib\_VideoColorABGR2RGB, 1294  
 color conversion —  
     mllib\_VideoColorARGB2RGB, 1299  
 color conversion —  
     mllib\_VideoColorRGB2ABGR, 1325  
 color conversion —  
     mllib\_VideoColorRGB2ARGB, 1326  
 color conversion (color channel merge) —  
     mllib\_VideoColorMerge2, 1318  
 color conversion (color channel merge) —  
     mllib\_VideoColorMerge2\_S16, 1319  
 color conversion (color channel merge) —  
     mllib\_VideoColorMerge3, 1320  
 color conversion (color channel merge) —  
     mllib\_VideoColorMerge3\_S16, 1321  
 color conversion (color channel merge) —  
     mllib\_VideoColorMerge4, 1322  
 color conversion (color channel merge) —  
     mllib\_VideoColorMerge4\_S16, 1323  
 color conversion (color channel split) —  
     mllib\_VideoColorSplit2, 1339  
 color conversion (color channel split) —  
     mllib\_VideoColorSplit2\_S16, 1340  
 color conversion (color channel split) —  
     mllib\_VideoColorSplit3, 1341  
 color conversion (color channel split) —  
     mllib\_VideoColorSplit3\_S16, 1342

color conversion (color channel split) —  
   mllib\_VideoColorSplit4, 1343  
 color conversion (color channel split) —  
   mllib\_VideoColorSplit4\_S16, 1344  
 color conversion using a 3x3 floating-point  
   matrix — mllib\_ImageColorConvert1, 289  
 color conversion using a 3x3 floating-point  
   matrix —  
   mllib\_ImageColorConvert1\_Fp, 290  
 color conversion using a 3x3 floating-point  
   matrix and a three-element offset —  
   mllib\_ImageColorConvert2, 291  
 color conversion using a 3x3 floating-point  
   matrix and a three-element offset —  
   mllib\_ImageColorConvert2\_Fp, 292  
 color convert UYV interleaved to ABGR  
   interleaved —  
   mllib\_VideoColorUYV444int\_to\_ABGRint, 1345  
 color convert UYV interleaved to ARGB  
   interleaved —  
   mllib\_VideoColorUYV444int\_to\_ARGBint, 1347  
 color convert UYVY interleaved to ABGR  
   interleaved —  
   mllib\_VideoColorUYVY422int\_to\_ABGRint, 1351  
 color convert UYVY interleaved to ARGB  
   interleaved —  
   mllib\_VideoColorUYVY422int\_to\_ARGBint, 1353  
 color convert YUV interleaved to ABGR  
   interleaved —  
   mllib\_VideoColorYUV444int\_to\_ABGRint, 1415  
 color convert YUV interleaved to ARGB  
   interleaved —  
   mllib\_VideoColorYUV444int\_to\_ARGBint, 1417  
 color convert YUV sequential to ABGR  
   interleaved —  
   mllib\_VideoColorYUV411seq\_to\_ABGRint, 1391  
 color convert YUV sequential to ABGR  
   interleaved —  
   mllib\_VideoColorYUV420seq\_to\_ABGRint, 1399  
 color convert YUV sequential to ABGR  
   interleaved —  
   mllib\_VideoColorYUV422seq\_to\_ABGRint, 1407  
 color convert YUV sequential to ABGR  
   interleaved —  
   mllib\_VideoColorYUV444seq\_to\_ABGRint, 1421  
 color convert YUV sequential to ARGB  
   interleaved —  
   mllib\_VideoColorYUV411seq\_to\_ARGBint, 1393  
 color convert YUV sequential to ARGB  
   interleaved —  
   mllib\_VideoColorYUV420seq\_to\_ARGBint, 1401  
 color convert YUV sequential to ARGB  
   interleaved —  
   mllib\_VideoColorYUV422seq\_to\_ARGBint, 1409  
 color convert YUV sequential to ARGB  
   interleaved —  
   mllib\_VideoColorYUV444seq\_to\_ARGBint, 1423  
 color convert YUYV interleaved to ABGR  
   interleaved —  
   mllib\_VideoColorYUYV422int\_to\_ABGRint, 1427  
 color convert YUYV interleaved to ARGB  
   interleaved —  
   mllib\_VideoColorYUYV422int\_to\_ARGBint, 1429  
 computes the absolute value of the image pixels  
   — mllib\_ImageAbs, 223  
 computes the absolute value of the image pixels  
   — mllib\_ImageAbs\_Fp, 224  
 computes the absolute value of the image pixels  
   — mllib\_ImageAbs\_Fp\_Inp, 225  
 computes the absolute value of the image  
   pixels, in place — mllib\_ImageAbs\_Inp, 226  
 computes the addition of two images on a  
   pixel-by-pixel basis — mllib\_ImageAdd, 227  
 computes the addition of two images on a  
   pixel-by-pixel basis —  
   mllib\_ImageAdd\_Fp, 228  
 computes the addition of two images on a  
   pixel-by-pixel basis —  
   mllib\_ImageAdd\_Fp\_Inp, 229  
 computes the addition of two images on a  
   pixel-by-pixel basis, in place —  
   mllib\_ImageAdd\_Inp, 230  
 computes the And of the first source image and  
   the Not of the second source image —  
   mllib\_ImageAndNot, 251  
 computes the And of the first source image and  
   the Not of the second source image, in place  
   — mllib\_ImageAndNot1\_Inp, 249  
 computes the And of the first source image and  
   the Not of the second source image, in place  
   — mllib\_ImageAndNot2\_Inp, 250  
 computes the And of two image, in place —  
   mllib\_ImageAnd\_Inp, 248  
 computes the And of two images —  
   mllib\_ImageAnd, 247

computes the exponent of the image pixels —  
     mllib\_ImageExp, 449  
 computes the exponent of the image pixels —  
     mllib\_ImageExp\_Fp, 450  
 computes the exponent of the image pixels —  
     mllib\_ImageExp\_Fp\_Inp, 451  
 computes the exponent of the image pixels —  
     mllib\_ImageExp\_Inp, 452  
 computes the multiplication of two images on a  
 pixel-by-pixel basis —  
     mllib\_ImageMul\_Fp, 603  
 computes the multiplication of two images on a  
 pixel-by-pixel basis —  
     mllib\_ImageMul\_Fp\_Inp, 604  
 computes the natural logarithm of the image  
 pixels — mllib\_ImageLog, 524  
 computes the natural logarithm of the image  
 pixels — mllib\_ImageLog\_Fp, 525  
 computes the natural logarithm of the image  
 pixels — mllib\_ImageLog\_Fp\_Inp, 526  
 computes the natural logarithm of the image  
 pixels, in place — mllib\_ImageLog\_Inp, 527  
 convert line spectral pair coefficients to linear  
 prediction coefficients —  
     mllib\_SignalLSP2LPC\_S16, 1097  
 convert line spectral pair coefficients to linear  
 prediction coefficients —  
     mllib\_SignalLSP2LPC\_S16\_AdP, 1097  
 convert ABGR interleaved to ARGB —  
     mllib\_VideoColorABGRint\_to\_ARGBint, 1295  
 convert BGR interleaved to ABGR interleaved  
 —  
     mllib\_VideoColorBGRint\_to\_ABGRint, 1301  
 convert BGRA interleaved to ABGR —  
     mllib\_VideoColorBGRAint\_to\_ABGRint, 1300  
 convert line spectral pair coefficients to linear  
 prediction coefficients —  
     mllib\_SignalLSP2LPC\_F32, 1095  
 convert linear prediction coefficients to cepstral  
 coefficients —  
     mllib\_SignalLPC2Cepstral\_F32, 1056  
 convert linear prediction coefficients to cepstral  
 coefficients —  
     mllib\_SignalLPC2Cepstral\_S16, 1058  
 convert linear prediction coefficients to cepstral  
 coefficients —  
     mllib\_SignalLPC2Cepstral\_S16\_AdP, 1060  
 convert linear prediction coefficients to line  
 spectral pair coefficients —  
     mllib\_SignalLPC2LSP\_F32, 1062  
 convert linear prediction coefficients to line  
 spectral pair coefficients —  
     mllib\_SignalLPC2LSP\_S16, 1064  
 convert RGB interleaved to ABGR interleaved  
 —  
     mllib\_VideoColorRGBint\_to\_ABGRint, 1332  
 convert RGB sequential to ABGR interleaved —  
     mllib\_VideoColorRGBseq\_to\_ABGRint, 1334  
 convert RGBA interleaved to ABGR —  
     mllib\_VideoColorRGBAint\_to\_ABGRint, 1331  
 convert RGBX interleaved to ABGR interleaved  
 —  
     mllib\_VideoColorRGBXint\_to\_ABGRint, 1336  
 convert RGBX interleaved to ARGB interleaved  
 —  
     mllib\_VideoColorRGBXint\_to\_ARGBint, 1338  
 convert UYV interleaved with subsampling —  
     mllib\_VideoColorUYV444int\_to\_UYVY422int, 1349  
 convert UYV interleaved with subsampling —  
     mllib\_VideoColorUYV444int\_to\_YUYV422int, 1350  
 convert XRGB interleaved to ABGR interleaved  
 —  
     mllib\_VideoColorXRGBint\_to\_ABGRint, 1355  
 convert XRGB interleaved to ARGB interleaved  
 —  
     mllib\_VideoColorXRGBint\_to\_ARGBint, 1356  
 convert YUV interleaved with subsampling —  
     mllib\_VideoColorYUV444int\_to\_UYVY422int, 1419  
 convert YUV interleaved with subsampling —  
     mllib\_VideoColorYUV444int\_to\_YUYV422int, 1420  
 convert YUV sequential to interleaved —  
     mllib\_VideoColorYUV411seq\_to\_UYVY422int, 1395  
 convert YUV sequential to interleaved —  
     mllib\_VideoColorYUV411seq\_to\_YUYV422int, 1397  
 convert YUV sequential to interleaved —  
     mllib\_VideoColorYUV420seq\_to\_UYVY422int, 1403  
 convert YUV sequential to interleaved —  
     mllib\_VideoColorYUV420seq\_to\_YUYV422int, 1405  
 convert YUV sequential to interleaved —  
     mllib\_VideoColorYUV422seq\_to\_UYVY422int, 1411  
 convert YUV sequential to interleaved —  
     mllib\_VideoColorYUV422seq\_to\_YUYV422int, 1413  
 convert YUV sequential to interleaved with  
 subsampling —  
     mllib\_VideoColorYUV444seq\_to\_UYVY422int, 1425

convert YUV sequential to interleaved with subsampling —  
     mllib\_VideoColorYUV444seq\_to\_YUYV422int, 1426  
 convolution kernel conversion —  
     mllib\_ImageConvKernelConvert, 401  
 copies a block from the reference block to the current block —  
     mllib\_VideoCopyRef\_S16\_U8\_16x16, 1435  
 copies a block from the reference block to the current block —  
     mllib\_VideoCopyRef\_S16\_U8\_16x8, 1435  
 copies a block from the reference block to the current block —  
     mllib\_VideoCopyRef\_S16\_U8\_8x16, 1435  
 copies a block from the reference block to the current block —  
     mllib\_VideoCopyRef\_S16\_U8\_8x4, 1435  
 copies a block from the reference block to the current block —  
     mllib\_VideoCopyRef\_S16\_U8\_8x8, 1435  
 copies an 8x8 block from the reference block to the current block —  
     mllib\_VideoCopyRef\_U8\_U8\_16x16, 1439  
 copies an 8x8 block from the reference block to the current block —  
     mllib\_VideoCopyRef\_U8\_U8\_16x8, 1439  
 copies an 8x8 block from the reference block to the current block —  
     mllib\_VideoCopyRef\_U8\_U8\_8x16, 1439  
 copies an 8x8 block from the reference block to the current block —  
     mllib\_VideoCopyRef\_U8\_U8\_8x4, 1439  
 copies an 8x8 block from the reference block to the current block —  
     mllib\_VideoCopyRef\_U8\_U8\_8x8, 1439  
 copies and averages a block from the reference block to the current block —  
     mllib\_VideoCopyRefAve\_U8\_U8\_16x16, 1431  
 copies and averages a block from the reference block to the current block —  
     mllib\_VideoCopyRefAve\_U8\_U8\_16x8, 1431  
 copies and averages a block from the reference block to the current block —  
     mllib\_VideoCopyRefAve\_U8\_U8\_8x16, 1431  
 copies and averages a block from the reference block to the current block —  
     mllib\_VideoCopyRefAve\_U8\_U8\_8x4, 1431  
 copies and averages a block from the reference block to the current block —  
     mllib\_VideoCopyRefAve\_U8\_U8\_8x8, 1431  
 copies a block from the reference block to the current block —  
     mllib\_VideoCopyRef\_S16\_U8, 1437  
 copies a block from the reference block to the current block —  
     mllib\_VideoCopyRef\_U8\_U8, 1441  
 copies and averages a block from the reference block to the current block —  
     mllib\_VideoCopyRefAve\_U8\_U8, 1433  
 copy a block of bytes — mllib\_memcpy, 845  
 copy a block of bytes — mllib\_memmove, 846  
 copy an area — mllib\_ImageCopyArea, 414  
 copy subimage —  
     mllib\_ImageCopySubimage, 417  
 copy with mask — mllib\_ImageCopyMask, 415  
 copy with mask, floating-point —  
     mllib\_ImageCopyMask\_Fp, 416  
 creates an interpolation table —  
     mllib\_ImageInterpTableCreate, 504  
 cross correlation —  
     mllib\_ImageCrossCorrel, 421  
 cross correlation —  
     mllib\_ImageCrossCorrel\_Fp, 422

## D

data type convert with shifting —  
     mllib\_SignalConvertShift\_F32S\_S16S, 867  
 data type convert with shifting —  
     mllib\_SignalConvertShift\_F32S\_S32S, 867  
 data type convert with shifting —  
     mllib\_SignalConvertShift\_F32S\_S8S, 867  
 data type convert with shifting —  
     mllib\_SignalConvertShift\_F32S\_U8S, 867  
 data type convert with shifting —  
     mllib\_SignalConvertShift\_F32\_S16, 867  
 data type convert with shifting —  
     mllib\_SignalConvertShift\_F32\_S32, 867  
 data type convert with shifting —  
     mllib\_SignalConvertShift\_F32\_S8, 867  
 data type convert with shifting —  
     mllib\_SignalConvertShift\_F32\_U8, 867  
 data type convert with shifting —  
     mllib\_SignalConvertShift\_S16S\_F32S\_Sat, 869

- data type convert with shifting —  
  - mllib\_SignalConvertShift\_S16S\_S32S\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_S16S\_S8S\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_S16S\_U8S\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_S16\_F32\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_S16\_S32\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_S16\_S8\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_S16\_U8\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_S32S\_F32S\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_S32S\_S16S\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_S32S\_S8S\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_S32S\_U8S\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_S32\_F32\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_S32\_S16\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_S32\_S8\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_S32\_U8\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_S8S\_F32S\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_S8S\_S16S\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_S8S\_S32S\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_S8S\_U8S\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_S8\_F32\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_S8\_S16\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_S8\_S32\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_S8\_U8\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_U8S\_F32S\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_U8S\_S16S\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_U8S\_S32S\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_U8S\_S8S\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_U8\_F32\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_U8\_S16\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_U8\_S32\_Sat, 869
- data type convert with shifting —  
  - mllib\_SignalConvertShift\_U8\_S8\_Sat, 869
- data type conversion —  
  - mllib\_ImageDataTypeConvert, 423
- deletes an interpolation table —  
  - mllib\_ImageInterpTableDelete, 506
- dequantization of forward Discrete Cosine Transform (DCT) coefficients —  
  - mllib\_VideoDeQuantizeInit\_S16, 1452
- dequantization of forward Discrete Cosine Transform (DCT) coefficients —  
  - mllib\_VideoDeQuantize\_S16, 1453
- division — mllib\_ImageDiv\_Fp, 441
- division, in place —  
  - mllib\_ImageDiv1\_Fp\_Inp, 431
- division, in place —  
  - mllib\_ImageDiv2\_Fp\_Inp, 432
- division by a constant, with shifting —  
  - mllib\_ImageDivConstShift, 439
- division by a constant, with shifting —  
  - mllib\_ImageDivConstShift\_Inp, 440
- division into a constant —  
  - mllib\_ImageConstDiv, 343
- division into a constant —  
  - mllib\_ImageConstDiv\_Fp, 344
- division into a constant —  
  - mllib\_ImageConstDiv\_Fp\_Inp, 345
- division into a constant —  
  - mllib\_ImageConstDiv\_Inp, 346
- division into a constant, with shifting —  
  - mllib\_ImageConstDivShift, 347
- division into a constant, with shifting —  
  - mllib\_ImageConstDivShift\_Inp, 348
- division with shifting —  
  - mllib\_ImageDivShift, 444

- division with shifting, in place —
  - mllib\_ImageDivShift1\_Inp, 442
- division with shifting, in place —
  - mllib\_ImageDivShift2\_Inp, 443
- Dolby AC-3 digital audio standard transformation —
  - mllib\_SignalIMDCTSplit\_D64, 1040
- Dolby AC-3 digital audio standard transformation —
  - mllib\_SignalIMDCTSplit\_F32, 1041
- Dolby AC-3 digital audio standard transformation —
  - mllib\_SignalIMDCT\_D64, 1038
- Dolby AC-3 digital audio standard transformation —
  - mllib\_SignalIMDCT\_F32, 1039
- down sampling rate conversion in JFIF —
  - mllib\_VideoDownSample420, 1454
- down sampling rate conversion in JFIF —
  - mllib\_VideoDownSample420\_S16, 1455
- down sampling rate conversion in JFIF —
  - mllib\_VideoDownSample422, 1456
- down sampling rate conversion in JFIF —
  - mllib\_VideoDownSample422\_S16, 1457
- draw arc — mllib\_GraphicsDrawArc\_32, 42
- draw arc — mllib\_GraphicsDrawArc\_8, 42
- draw arc in Xor mode —
  - mllib\_GraphicsDrawArc\_X\_32, 44
- draw arc in Xor mode —
  - mllib\_GraphicsDrawArc\_X\_8, 44
- draw arc with antialiasing —
  - mllib\_GraphicsDrawArc\_A\_32, 43
- draw arc with antialiasing —
  - mllib\_GraphicsDrawArc\_A\_8, 43
- draw circle — mllib\_GraphicsDrawCircle\_32, 45
- draw circle — mllib\_GraphicsDrawCircle\_8, 45
- draw circle in Xor mode —
  - mllib\_GraphicsDrawCircle\_X\_32, 47
- draw circle in Xor mode —
  - mllib\_GraphicsDrawCircle\_X\_8, 47
- draw circle with antialiasing —
  - mllib\_GraphicsDrawCircle\_A\_32, 46
- draw circle with antialiasing —
  - mllib\_GraphicsDrawCircle\_A\_8, 46
- draw ellipse —
  - mllib\_GraphicsDrawEllipse\_32, 48
- draw ellipse —
  - mllib\_GraphicsDrawEllipse\_8, 48
- draw ellipse in Xor mode —
  - mllib\_GraphicsDrawEllipse\_X\_32, 50
- draw ellipse in Xor mode —
  - mllib\_GraphicsDrawEllipse\_X\_8, 50
- draw ellipse with antialiasing —
  - mllib\_GraphicsDrawEllipse\_A\_32, 49
- draw ellipse with antialiasing —
  - mllib\_GraphicsDrawEllipse\_A\_8, 49
- draw filled arc — mllib\_GraphicsFillArc\_32, 159
- draw filled arc — mllib\_GraphicsFillArc\_8, 159
- draw filled arc in Xor mode —
  - mllib\_GraphicsFillArc\_X\_32, 161
- draw filled arc in Xor mode —
  - mllib\_GraphicsFillArc\_X\_8, 161
- draw filled arc with antialiasing —
  - mllib\_GraphicsFillArc\_A\_32, 160
- draw filled arc with antialiasing —
  - mllib\_GraphicsFillArc\_A\_8, 160
- draw filled circle —
  - mllib\_GraphicsFillCircle\_32, 162
- draw filled circle —
  - mllib\_GraphicsFillCircle\_8, 162
- draw filled circle in Xor mode —
  - mllib\_GraphicsFillCircle\_X\_32, 164
- draw filled circle in Xor mode —
  - mllib\_GraphicsFillCircle\_X\_8, 164
- draw filled circle with antialiasing —
  - mllib\_GraphicsFillCircle\_A\_32, 163
- draw filled circle with antialiasing —
  - mllib\_GraphicsFillCircle\_A\_8, 163
- draw filled ellipse —
  - mllib\_GraphicsFillEllipse\_32, 165
- draw filled ellipse —
  - mllib\_GraphicsFillEllipse\_8, 165
- draw filled ellipse in Xor mode —
  - mllib\_GraphicsFillEllipse\_X\_32, 167
- draw filled ellipse in Xor mode —
  - mllib\_GraphicsFillEllipse\_X\_8, 167
- draw filled ellipse with antialiasing —
  - mllib\_GraphicsFillEllipse\_A\_32, 166
- draw filled ellipse with antialiasing —
  - mllib\_GraphicsFillEllipse\_A\_8, 166
- draw filled polygon —
  - mllib\_GraphicsFillPolygon\_32, 168
- draw filled polygon —
  - mllib\_GraphicsFillPolygon\_8, 168
- draw filled polygon in Xor mode —
  - mllib\_GraphicsFillPolygon\_X\_32, 175

draw filled polygon in Xor mode —  
 mlib\_GraphicsFillPolygon\_X\_8, 175

draw filled polygon with antialiasing —  
 mlib\_GraphicsFillPolygon\_A\_32, 169

draw filled polygon with antialiasing —  
 mlib\_GraphicsFillPolygon\_A\_8, 169

draw filled polygon with antialiasing and  
 Gouraud shading —  
 mlib\_GraphicsFillPolygon\_AG\_32, 170

draw filled polygon with antialiasing and  
 Gouraud shading —  
 mlib\_GraphicsFillPolygon\_AG\_8, 170

draw filled polygon with antialiasing and Z  
 buffering —  
 mlib\_GraphicsFillPolygon\_AZ\_32, 172

draw filled polygon with antialiasing and Z  
 buffering —  
 mlib\_GraphicsFillPolygon\_AZ\_8, 172

draw filled polygon with antialiasing, Gouraud  
 shading, and Z buffering —  
 mlib\_GraphicsFillPolygon\_AGZ\_32, 171

draw filled polygon with antialiasing, Gouraud  
 shading, and Z buffering —  
 mlib\_GraphicsFillPolygon\_AGZ\_8, 171

draw filled polygon with Gouraud shading —  
 mlib\_GraphicsFillPolygon\_G\_32, 173

draw filled polygon with Gouraud shading —  
 mlib\_GraphicsFillPolygon\_G\_8, 173

draw filled polygon with Gouraud shading and  
 Z buffering —  
 mlib\_GraphicsFillPolygon\_GZ\_32, 174

draw filled polygon with Gouraud shading and  
 Z buffering —  
 mlib\_GraphicsFillPolygon\_GZ\_8, 174

draw filled polygon with Z buffering —  
 mlib\_GraphicsFillPolygon\_Z\_32, 176

draw filled polygon with Z buffering —  
 mlib\_GraphicsFillPolygon\_Z\_8, 176

draw filled rectangle —  
 mlib\_GraphicsFillRectangle\_32, 177

draw filled rectangle —  
 mlib\_GraphicsFillRectangle\_8, 177

draw filled rectangle in Xor mode —  
 mlib\_GraphicsFillRectangle\_X\_32, 178

draw filled rectangle in Xor mode —  
 mlib\_GraphicsFillRectangle\_X\_8, 178

draw filled triangle —  
 mlib\_GraphicsFillTriangle\_32, 179

draw filled triangle —  
 mlib\_GraphicsFillTriangle\_8, 179

draw filled triangle in Xor mode —  
 mlib\_GraphicsFillTriangle\_X\_32, 218

draw filled triangle in Xor mode —  
 mlib\_GraphicsFillTriangle\_X\_8, 218

draw filled triangle set in Xor mode where all  
 members of the set have a common vertex —  
 mlib\_GraphicsFillTriangleFanSet\_X\_32, 194

draw filled triangle set in Xor mode where all  
 members of the set have a common vertex —  
 mlib\_GraphicsFillTriangleFanSet\_X\_8, 194

draw filled triangle set in Xor mode where each  
 member can have different vertices —  
 mlib\_GraphicsFillTriangleSet\_X\_32, 207

draw filled triangle set in Xor mode where each  
 member can have different vertices —  
 mlib\_GraphicsFillTriangleSet\_X\_8, 207

draw filled triangle set in Xor mode where the  
 first side of each member is common to the  
 second side of the previous member —  
 mlib\_GraphicsFillTriangleStripSet\_X\_32, 216

draw filled triangle set in Xor mode where the  
 first side of each member is common to the  
 second side of the previous member —  
 mlib\_GraphicsFillTriangleStripSet\_X\_8, 216

draw filled triangle set where all members of  
 the set have a common vertex —  
 mlib\_GraphicsFillTriangleFanSet\_32, 187

draw filled triangle set where all members of  
 the set have a common vertex —  
 mlib\_GraphicsFillTriangleFanSet\_8, 187

draw filled triangle set where each member can  
 have different vertices —  
 mlib\_GraphicsFillTriangleSet\_32, 200

draw filled triangle set where each member can  
 have different vertices —  
 mlib\_GraphicsFillTriangleSet\_8, 200

draw filled triangle set where the first side of  
 each member is common to the second side  
 of the previous member —  
 mlib\_GraphicsFillTriangleStripSet\_32, 209

draw filled triangle set where the first side of  
 each member is common to the second side  
 of the previous member —  
 mlib\_GraphicsFillTriangleStripSet\_8, 209



draw filled triangle set with antialiasing and gouraud shading, where all members of the set have a common vertex —  
 mlib\_GraphicsFillTriangleFanSet\_AG\_32, 189

draw filled triangle set with antialiasing and gouraud shading, where all members of the set have a common vertex —  
 mlib\_GraphicsFillTriangleFanSet\_AG\_8, 189

draw filled triangle set with antialiasing and Gouraud shading, where each member can have different vertices —  
 mlib\_GraphicsFillTriangleSet\_AG\_32, 202

draw filled triangle set with antialiasing and Gouraud shading, where each member can have different vertices —  
 mlib\_GraphicsFillTriangleSet\_AG\_8, 202

draw filled triangle set with antialiasing and gouraud shading, where the first side of each member is common to the second side of the previous member —  
 mlib\_GraphicsFillTriangleStripSet\_AG\_32, 211

draw filled triangle set with antialiasing and gouraud shading, where the first side of each member is common to the second side of the previous member —  
 mlib\_GraphicsFillTriangleStripSet\_AG\_8, 211

draw filled triangle set with antialiasing and Z buffering, where all members of the set have a common vertex —  
 mlib\_GraphicsFillTriangleFanSet\_AZ\_32, 191

draw filled triangle set with antialiasing and Z buffering, where all members of the set have a common vertex —  
 mlib\_GraphicsFillTriangleFanSet\_AZ\_8, 191

draw filled triangle set with antialiasing and Z buffering, where each member can have different vertices —  
 mlib\_GraphicsFillTriangleSet\_AZ\_32, 204

draw filled triangle set with antialiasing and Z buffering, where each member can have different vertices —  
 mlib\_GraphicsFillTriangleSet\_AZ\_8, 204

draw filled triangle set with antialiasing and Z buffering, where the first side of each member is common to the second side of the previous member —  
 mlib\_GraphicsFillTriangleStripSet\_AZ\_32, 213

draw filled triangle set with antialiasing and Z buffering, where the first side of each member is common to the second side of the previous member —  
 mlib\_GraphicsFillTriangleStripSet\_AZ\_8, 213

draw filled triangle set with antialiasing where all members of the set have a common vertex —  
 mlib\_GraphicsFillTriangleFanSet\_A\_32, 188

draw filled triangle set with antialiasing where all members of the set have a common vertex —  
 mlib\_GraphicsFillTriangleFanSet\_A\_8, 188

draw filled triangle set with antialiasing where each member can have different vertices —  
 mlib\_GraphicsFillTriangleSet\_A\_32, 201

draw filled triangle set with antialiasing where each member can have different vertices —  
 mlib\_GraphicsFillTriangleSet\_A\_8, 201

draw filled triangle set with antialiasing where the first side of each member is common to the second side of the previous member —  
 mlib\_GraphicsFillTriangleStripSet\_A\_32, 210

draw filled triangle set with antialiasing where the first side of each member is common to the second side of the previous member —  
 mlib\_GraphicsFillTriangleStripSet\_A\_8, 210

draw filled triangle set with antialiasing, Gouraud shading, and Z buffering, where all members of the set have a common vertex —  
 mlib\_GraphicsFillTriangleFanSet\_AGZ\_32, 190

draw filled triangle set with antialiasing, Gouraud shading, and Z buffering, where all members of the set have a common vertex —  
 mlib\_GraphicsFillTriangleFanSet\_AGZ\_8, 190

draw filled triangle set with antialiasing, Gouraud shading, and Z buffering, where each member can have different vertices —  
 mlib\_GraphicsFillTriangleSet\_AGZ\_32, 203

draw filled triangle set with antialiasing, Gouraud shading, and Z buffering, where each member can have different vertices —  
 mlib\_GraphicsFillTriangleSet\_AGZ\_8, 203

draw filled triangle set with antialiasing, Gouraud shading, and Z buffering, where the first side of each member is common to the second side of the previous member —  
 mlib\_GraphicsFillTriangleStripSet\_AGZ, 212

draw filled triangle set with antialiasing, Gouraud shading, and Z buffering, where the first side of each member is common to the second side of the previous member — `mllib_GraphicsFillTriangleStripSet_AGZ_32`, 212

draw filled triangle set with antialiasing, Gouraud shading, and Z buffering, where the first side of each member is common to the second side of the previous member — `mllib_GraphicsFillTriangleStripSet_AGZ_8`, 212

draw filled triangle set with Gouraud shading and Z buffering, where all members of the set have a common vertex — `mllib_GraphicsFillTriangleFanSet_GZ_32`, 193

draw filled triangle set with Gouraud shading and Z buffering, where all members of the set have a common vertex — `mllib_GraphicsFillTriangleFanSet_GZ_8`, 193

draw filled triangle set with Gouraud shading and Z buffering, where each member can have different vertices — `mllib_GraphicsFillTriangleSet_GZ_32`, 206

draw filled triangle set with Gouraud shading and Z buffering, where each member can have different vertices — `mllib_GraphicsFillTriangleSet_GZ_8`, 206

draw filled triangle set with Gouraud shading and Z buffering, where the first side of each member is common to the second side of the previous member — `mllib_GraphicsFillTriangleStripSet_GZ_32`, 215

draw filled triangle set with Gouraud shading and Z buffering, where the first side of each member is common to the second side of the previous member — `mllib_GraphicsFillTriangleStripSet_GZ_8`, 215

draw filled triangle set with Gouraud shading where all members of the set have a common vertex — `mllib_GraphicsFillTriangleFanSet_G_32`, 192

draw filled triangle set with Gouraud shading where all members of the set have a common vertex — `mllib_GraphicsFillTriangleFanSet_G_8`, 192

draw filled triangle set with Gouraud shading where each member can have different vertices — `mllib_GraphicsFillTriangleSet_G_32`, 205

draw filled triangle set with Gouraud shading where each member can have different vertices — `mllib_GraphicsFillTriangleSet_G_8`, 205

draw filled triangle set with Gouraud shading where the first side of each member is common to the second side of the previous member — `mllib_GraphicsFillTriangleStripSet_G_32`, 214

draw filled triangle set with Gouraud shading where the first side of each member is common to the second side of the previous member — `mllib_GraphicsFillTriangleStripSet_G_8`, 214

draw filled triangle set with Z buffering where all members of the set have a common vertex — `mllib_GraphicsFillTriangleFanSet_Z_32`, 195

draw filled triangle set with Z buffering where all members of the set have a common vertex — `mllib_GraphicsFillTriangleFanSet_Z_8`, 195

draw filled triangle set with Z buffering where each member can have different vertices — `mllib_GraphicsFillTriangleSet_Z_32`, 208

draw filled triangle set with Z buffering where each member can have different vertices — `mllib_GraphicsFillTriangleSet_Z_8`, 208

draw filled triangle set with Z buffering where the first side of each member is common to the second side of the previous member — `mllib_GraphicsFillTriangleStripSet_Z_32`, 217

draw filled triangle set with Z buffering where the first side of each member is common to the second side of the previous member — `mllib_GraphicsFillTriangleStripSet_Z_8`, 217

draw filled triangle with antialiasing — `mllib_GraphicsFillTriangle_A_32`, 180

draw filled triangle with antialiasing — `mllib_GraphicsFillTriangle_A_8`, 180

draw filled triangle with antialiasing and Z buffering — `mllib_GraphicsFillTriangle_AZ_32`, 185

draw filled triangle with antialiasing and Z buffering — `mllib_GraphicsFillTriangle_AZ_8`, 185

draw filled triangle with antialiasing, Gouraud shading, and Z buffering —  
 mlib\_GraphicsFillTriangle\_AGZ\_32, 183

draw filled triangle with antialiasing, Gouraud shading, and Z buffering —  
 mlib\_GraphicsFillTriangle\_AGZ\_8, 183

draw filled triangle with Gouraud shading —  
 mlib\_GraphicsFillTriangle\_G\_32, 196

draw filled triangle with Gouraud shading —  
 mlib\_GraphicsFillTriangle\_G\_8, 196

draw filled triangle with Gouraud shading and Z buffering —  
 mlib\_GraphicsFillTriangle\_GZ\_32, 198

draw filled triangle with Gouraud shading and Z buffering —  
 mlib\_GraphicsFillTriangle\_GZ\_8, 198

draw filled triangle with Z buffering —  
 mlib\_GraphicsFillTriangle\_Z\_32, 220

draw filled triangle with Z buffering —  
 mlib\_GraphicsFillTriangle\_Z\_8, 220

draw line — mlib\_GraphicsDrawLine\_32, 51

draw line — mlib\_GraphicsDrawLine\_8, 51

draw line in Xor mode —  
 mlib\_GraphicsDrawLine\_X\_32, 87

draw line in Xor mode —  
 mlib\_GraphicsDrawLine\_X\_8, 87

draw line set in Xor mode where all members of the set have one common end point —  
 mlib\_GraphicsDrawLineFanSet\_X\_32, 64

draw line set in Xor mode where all members of the set have one common end point —  
 mlib\_GraphicsDrawLineFanSet\_X\_8, 64

draw line set in Xor mode where each member can have different end points —  
 mlib\_GraphicsDrawLineSet\_X\_32, 76

draw line set in Xor mode where each member can have different end points —  
 mlib\_GraphicsDrawLineSet\_X\_8, 76

draw line set in Xor mode where each member of the set starts at the point where the previous member ended —  
 mlib\_GraphicsDrawLineStripSet\_X\_32, 85

draw line set in Xor mode where each member of the set starts at the point where the previous member ended —  
 mlib\_GraphicsDrawLineStripSet\_X\_8, 85

draw line set where all members of the set have one common end point —  
 mlib\_GraphicsDrawLineFanSet\_32, 57

draw line set where all members of the set have one common end point —  
 mlib\_GraphicsDrawLineFanSet\_8, 57

draw line set where each member can have different end points —  
 mlib\_GraphicsDrawLineSet\_32, 69

draw line set where each member can have different end points —  
 mlib\_GraphicsDrawLineSet\_8, 69

draw line set where each member of the set starts at the point where the previous member ended —  
 mlib\_GraphicsDrawLineStripSet\_32, 78

draw line set where each member of the set starts at the point where the previous member ended —  
 mlib\_GraphicsDrawLineStripSet\_8, 78

draw line set with antialiasing and Gouraud shading where each member can have different end points —  
 mlib\_GraphicsDrawLineSet\_AG\_32, 71

draw line set with antialiasing and Gouraud shading where each member can have different end points —  
 mlib\_GraphicsDrawLineSet\_AG\_8, 71

draw line set with antialiasing and gouraud shading where each member of the set starts at the point where the previous member ended —  
 mlib\_GraphicsDrawLineStripSet\_AG\_32, 80

draw line set with antialiasing and gouraud shading where each member of the set starts at the point where the previous member ended —  
 mlib\_GraphicsDrawLineStripSet\_AG\_8, 80

draw line set with antialiasing and gouraud shading, where all members of the set have one common end point —  
 mlib\_GraphicsDrawLineFanSet\_AG\_32, 59

draw line set with antialiasing and gouraud shading, where all members of the set have one common end point —  
 mlib\_GraphicsDrawLineFanSet\_AG\_8, 59

draw line set with antialiasing and Z buffering, where all members of the set have one common end point —  
 mlib\_GraphicsDrawLineFanSet\_AZ\_32, 61

draw line set with antialiasing and Z buffering, where all members of the set have one common end point —  
 mlib\_GraphicsDrawLineFanSet\_AZ\_8, 61

draw line set with antialiasing and Z buffering, where each member can have different end points —  
 mlib\_GraphicsDrawLineSet\_AZ\_32, 73

draw line set with antialiasing and Z buffering, where each member can have different end points —  
 mlib\_GraphicsDrawLineSet\_AZ\_8, 73

draw line set with antialiasing and Z buffering, where each member of the set starts at the point where the previous member ended —  
 mlib\_GraphicsDrawLineStripSet\_AZ\_32, 82

draw line set with antialiasing and Z buffering, where each member of the set starts at the point where the previous member ended —  
 mlib\_GraphicsDrawLineStripSet\_AZ\_8, 82

draw line set with antialiasing where each member can have different end points —  
 mlib\_GraphicsDrawLineSet\_A\_32, 70

draw line set with antialiasing where each member can have different end points —  
 mlib\_GraphicsDrawLineSet\_A\_8, 70

draw line set with antialiasing where each member of the set starts at the point where the previous member ended —  
 mlib\_GraphicsDrawLineStripSet\_A\_32, 79

draw line set with antialiasing where each member of the set starts at the point where the previous member ended —  
 mlib\_GraphicsDrawLineStripSet\_A\_8, 79

draw line set with antialiasing, Gouraud shading, and Z buffering, where all members of the set have one common end point —  
 mlib\_GraphicsDrawLineFanSet\_AGZ\_32, 60

draw line set with antialiasing, Gouraud shading, and Z buffering, where all members of the set have one common end point —  
 mlib\_GraphicsDrawLineFanSet\_AGZ\_8, 60

draw line set with antialiasing, Gouraud shading, and Z buffering, where each member can have different end points —  
 mlib\_GraphicsDrawLineSet\_AGZ\_32, 72

draw line set with antialiasing, Gouraud shading, and Z buffering, where each member can have different end points —  
 mlib\_GraphicsDrawLineSet\_AGZ\_8, 72

draw line set with antialiasing, Gouraud shading, and Z buffering, where each member of the set starts at the point where the previous member ended —  
 mlib\_GraphicsDrawLineStripSet\_AGZ\_32, 81

draw line set with antialiasing, Gouraud shading, and Z buffering, where each member of the set starts at the point where the previous member ended —  
 mlib\_GraphicsDrawLineStripSet\_AGZ\_8, 81

draw line set with antialiasing, where all members of the set have one common end point —  
 mlib\_GraphicsDrawLineFanSet\_A\_32, 58

draw line set with antialiasing, where all members of the set have one common end point —  
 mlib\_GraphicsDrawLineFanSet\_A\_8, 58

draw line set with Gouraud shading and Z buffering, where all members of the set have one common end point —  
 mlib\_GraphicsDrawLineFanSet\_GZ\_32, 63

draw line set with Gouraud shading and Z buffering, where all members of the set have one common end point —  
 mlib\_GraphicsDrawLineFanSet\_GZ\_8, 63

draw line set with Gouraud shading and Z buffering, where each member can have different end points —  
 mlib\_GraphicsDrawLineSet\_GZ\_32, 75

draw line set with Gouraud shading and Z buffering, where each member can have different end points —  
 mlib\_GraphicsDrawLineSet\_GZ\_8, 75

draw line set with Gouraud shading and Z buffering, where each member of the set starts at the point where the previous member ended —  
 mlib\_GraphicsDrawLineStripSet\_GZ\_32, 84

draw line set with Gouraud shading and Z buffering, where each member of the set starts at the point where the previous member ended —  
 mlib\_GraphicsDrawLineStripSet\_GZ\_8, 84

draw line set with Gouraud shading where each member can have different end points —  
 mlib\_GraphicsDrawLineSet\_G\_32, 74

draw line set with Gouraud shading where each member can have different end points —  
 mlib\_GraphicsDrawLineSet\_G\_8, 74

draw line set with Gouraud shading, where all members of the set have one common end point —  
 mlib\_GraphicsDrawLineFanSet\_G\_32, 62

draw line set with Gouraud shading, where all members of the set have one common end point —  
 mlib\_GraphicsDrawLineFanSet\_G\_8, 62

draw line set with Gouraud shading, where each member of the set starts at the point where the previous member ended —  
 mlib\_GraphicsDrawLineStripSet\_G\_32, 83

draw line set with Gouraud shading, where each member of the set starts at the point where the previous member ended —  
 mlib\_GraphicsDrawLineStripSet\_G\_8, 83

draw line set with Z buffering where all members of the set have one common end point —  
 mlib\_GraphicsDrawLineFanSet\_Z\_32, 65

draw line set with Z buffering where all members of the set have one common end point —  
 mlib\_GraphicsDrawLineFanSet\_Z\_8, 65

draw line set with Z buffering where each member can have different end points —  
 mlib\_GraphicsDrawLineSet\_Z\_32, 77

draw line set with Z buffering where each member can have different end points —  
 mlib\_GraphicsDrawLineSet\_Z\_8, 77

draw line set with Z buffering where each member of the set starts at the point where the previous member ended —  
 mlib\_GraphicsDrawLineStripSet\_Z\_32, 86

draw line set with Z buffering where each member of the set starts at the point where the previous member ended —  
 mlib\_GraphicsDrawLineStripSet\_Z\_8, 86

draw line with antialiasing —  
 mlib\_GraphicsDrawLine\_A\_32, 52

draw line with antialiasing —  
 mlib\_GraphicsDrawLine\_A\_8, 52

draw line with antialiasing and Gouraud shading —  
 mlib\_GraphicsDrawLine\_AG\_32, 53

draw line with antialiasing and Gouraud shading —  
 mlib\_GraphicsDrawLine\_AG\_8, 53

draw line with antialiasing and Gouraud shading —  
 mlib\_GraphicsDrawPolygon\_AG\_32, 95

draw line with antialiasing and Gouraud shading —  
 mlib\_GraphicsDrawPolygon\_AG\_8, 95

draw line with antialiasing and Gouraud shading —  
 mlib\_GraphicsDrawPolyline\_AG\_32, 104

draw line with antialiasing and Gouraud shading —  
 mlib\_GraphicsDrawPolyline\_AG\_8, 104

draw line with antialiasing and Gouraud shading —  
 mlib\_GraphicsFillTriangle\_AG\_32, 181

draw line with antialiasing and Gouraud shading —  
 mlib\_GraphicsFillTriangle\_AG\_8, 181

draw line with antialiasing and Z buffering —  
 mlib\_GraphicsDrawLine\_AZ\_32, 56

draw line with antialiasing and Z buffering —  
 mlib\_GraphicsDrawLine\_AZ\_8, 56

draw line with antialiasing, Gouraud shading, and Z buffering —  
 mlib\_GraphicsDrawLine\_AGZ\_32, 54

draw line with antialiasing, Gouraud shading, and Z buffering —  
 mlib\_GraphicsDrawLine\_AGZ\_8, 54

draw line with Gouraud shading —  
 mlib\_GraphicsDrawLine\_G\_32, 66

draw line with Gouraud shading —  
 mlib\_GraphicsDrawLine\_G\_8, 66

draw line with Gouraud shading and Z buffering —  
     mllib\_GraphicsDrawLine\_GZ\_32, 67  
 draw line with Gouraud shading and Z buffering —  
     mllib\_GraphicsDrawLine\_GZ\_8, 67  
 draw line with Z buffering —  
     mllib\_GraphicsDrawLine\_Z\_32, 88  
 draw line with Z buffering —  
     mllib\_GraphicsDrawLine\_Z\_8, 88  
 draw point — mllib\_GraphicsDrawPoint\_32, 89  
 draw point — mllib\_GraphicsDrawPoint\_8, 89  
 draw point in Xor mode —  
     mllib\_GraphicsDrawPoint\_X\_32, 92  
 draw point in Xor mode —  
     mllib\_GraphicsDrawPoint\_X\_8, 92  
 draw point set in Xor mode where each member can be a different point —  
     mllib\_GraphicsDrawPointSet\_X\_32, 91  
 draw point set in Xor mode where each member can be a different point —  
     mllib\_GraphicsDrawPointSet\_X\_8, 91  
 draw point set where each member can be a different point —  
     mllib\_GraphicsDrawPointSet\_32, 90  
 draw point set where each member can be a different point —  
     mllib\_GraphicsDrawPointSet\_8, 90  
 draw polygon —  
     mllib\_GraphicsDrawPolygon\_32, 93  
 draw polygon —  
     mllib\_GraphicsDrawPolygon\_8, 93  
 draw polygon in Xor mode —  
     mllib\_GraphicsDrawPolygon\_X\_32, 100  
 draw polygon in Xor mode —  
     mllib\_GraphicsDrawPolygon\_X\_8, 100  
 draw polygon with antialiasing —  
     mllib\_GraphicsDrawPolygon\_A\_32, 94  
 draw polygon with antialiasing —  
     mllib\_GraphicsDrawPolygon\_A\_8, 94  
 draw polygon with antialiasing and Z buffering —  
     mllib\_GraphicsDrawPolygon\_AZ\_32, 97  
 draw polygon with antialiasing and Z buffering —  
     mllib\_GraphicsDrawPolygon\_AZ\_8, 97  
 draw polygon with antialiasing, Gouraud shading, and Z buffering —  
     mllib\_GraphicsDrawPolygon\_AGZ\_32, 96  
 draw polygon with antialiasing, Gouraud shading, and Z buffering —  
     mllib\_GraphicsDrawPolygon\_AGZ\_8, 96  
 draw polygon with Gouraud shading —  
     mllib\_GraphicsDrawPolygon\_G\_32, 98  
 draw polygon with Gouraud shading —  
     mllib\_GraphicsDrawPolygon\_G\_8, 98  
 draw polygon with Gouraud shading and Z buffering —  
     mllib\_GraphicsDrawPolygon\_GZ\_32, 99  
 draw polygon with Gouraud shading and Z buffering —  
     mllib\_GraphicsDrawPolygon\_GZ\_8, 99  
 draw polygon with Z buffering —  
     mllib\_GraphicsDrawPolygon\_Z\_32, 101  
 draw polygon with Z buffering —  
     mllib\_GraphicsDrawPolygon\_Z\_8, 101  
 draw polyline —  
     mllib\_GraphicsDrawPolyline\_32, 102  
 draw polyline —  
     mllib\_GraphicsDrawPolyline\_8, 102  
 draw polyline in Xor mode —  
     mllib\_GraphicsDrawPolyline\_X\_32, 109  
 draw polyline in Xor mode —  
     mllib\_GraphicsDrawPolyline\_X\_8, 109  
 draw polyline with antialiasing —  
     mllib\_GraphicsDrawPolyline\_A\_32, 103  
 draw polyline with antialiasing —  
     mllib\_GraphicsDrawPolyline\_A\_8, 103  
 draw polyline with antialiasing and Z buffering —  
     mllib\_GraphicsDrawPolyline\_AZ\_32, 106  
 draw polyline with antialiasing and Z buffering —  
     mllib\_GraphicsDrawPolyline\_AZ\_8, 106  
 draw polyline with antialiasing, Gouraud shading, and Z buffering —  
     mllib\_GraphicsDrawPolyline\_AGZ\_32, 105  
 draw polyline with antialiasing, Gouraud shading, and Z buffering —  
     mllib\_GraphicsDrawPolyline\_AGZ\_8, 105  
 draw polyline with Gouraud shading —  
     mllib\_GraphicsDrawPolyline\_G\_32, 107  
 draw polyline with Gouraud shading —  
     mllib\_GraphicsDrawPolyline\_G\_8, 107  
 draw polyline with Gouraud shading and Z buffering —  
     mllib\_GraphicsDrawPolyline\_GZ\_32, 108

draw polyline with Gouraud shading and Z buffering —  
 mlib\_GraphicsDrawPolyline\_GZ\_8, 108

draw polyline with Z buffering —  
 mlib\_GraphicsDrawPolyline\_Z\_32, 110

draw polyline with Z buffering —  
 mlib\_GraphicsDrawPolyline\_Z\_8, 110

draw polypoint —  
 mlib\_GraphicsDrawPolypoint\_32, 111

draw polypoint —  
 mlib\_GraphicsDrawPolypoint\_8, 111

draw polypoint in Xor mode —  
 mlib\_GraphicsDrawPolypoint\_X\_32, 112

draw polypoint in Xor mode —  
 mlib\_GraphicsDrawPolypoint\_X\_8, 112

draw rectangle —  
 mlib\_GraphicsDrawRectangle\_32, 113

draw rectangle —  
 mlib\_GraphicsDrawRectangle\_8, 113

draw rectangle in Xor mode —  
 mlib\_GraphicsDrawRectangle\_X\_32, 114

draw rectangle in Xor mode —  
 mlib\_GraphicsDrawRectangle\_X\_8, 114

draw triangle —  
 mlib\_GraphicsDrawTriangle\_32, 115

draw triangle —  
 mlib\_GraphicsDrawTriangle\_8, 115

draw triangle in Xor mode —  
 mlib\_GraphicsDrawTriangle\_X\_32, 155

draw triangle in Xor mode —  
 mlib\_GraphicsDrawTriangle\_X\_8, 155

draw triangle set in Xor mode where all members of the set have a common vertex —  
 mlib\_GraphicsDrawTriangleFanSet\_X\_32, 130

draw triangle set in Xor mode where all members of the set have a common vertex —  
 mlib\_GraphicsDrawTriangleFanSet\_X\_8, 130

draw triangle set in Xor mode where each member can have different vertices —  
 mlib\_GraphicsDrawTriangleSet\_X\_32, 143

draw triangle set in Xor mode where each member can have different vertices —  
 mlib\_GraphicsDrawTriangleSet\_X\_8, 143

draw triangle set in Xor mode where the first side of each member is common to the second side of the previous member —  
 mlib\_GraphicsDrawTriangleStripSet\_X\_32, 153

draw triangle set in Xor mode where the first side of each member is common to the second side of the previous member —  
 mlib\_GraphicsDrawTriangleStripSet\_X\_8, 153

draw triangle set where all members of the set have a common vertex —  
 mlib\_GraphicsDrawTriangleFanSet\_32, 123

draw triangle set where all members of the set have a common vertex —  
 mlib\_GraphicsDrawTriangleFanSet\_8, 123

draw triangle set where each member can have different vertices —  
 mlib\_GraphicsDrawTriangleSet\_32, 136

draw triangle set where each member can have different vertices —  
 mlib\_GraphicsDrawTriangleSet\_8, 136

draw triangle set where the first side of each member is common to the second side of the previous member —  
 mlib\_GraphicsDrawTriangleStripSet\_32, 145

draw triangle set where the first side of each member is common to the second side of the previous member —  
 mlib\_GraphicsDrawTriangleStripSet\_8, 145

draw triangle set with antialiasing and gouraud shading —  
 mlib\_GraphicsDrawTriangleStripSet\_AG\_32, 147

draw triangle set with antialiasing and gouraud shading —  
 mlib\_GraphicsDrawTriangleStripSet\_AG\_8, 147

draw triangle set with antialiasing and gouraud shading, where all members of the set have a common vertex —  
 mlib\_GraphicsDrawTriangleFanSet\_AG\_32, 125

draw triangle set with antialiasing and gouraud shading, where all members of the set have a common vertex —  
 mlib\_GraphicsDrawTriangleFanSet\_AG\_8, 125

draw triangle set with antialiasing and Gouraud shading, where each member can have different vertices —  
 mlib\_GraphicsDrawTriangleSet\_AG\_32, 138

draw triangle set with antialiasing and Gouraud shading, where each member can have different vertices —  
 mlib\_GraphicsDrawTriangleSet\_AG\_8, 138

draw triangle set with antialiasing and Z buffering, where all members of the set have a common vertex —  
 mlib\_GraphicsDrawTriangleFanSet\_AZ\_32, 127

draw triangle set with antialiasing and Z buffering, where all members of the set have a common vertex —  
 mlib\_GraphicsDrawTriangleFanSet\_AZ\_8, 127

draw triangle set with antialiasing and Z buffering, where each member can have different vertices —  
 mlib\_GraphicsDrawTriangleSet\_AZ\_32, 140

draw triangle set with antialiasing and Z buffering, where each member can have different vertices —  
 mlib\_GraphicsDrawTriangleSet\_AZ\_8, 140

draw triangle set with antialiasing and Z buffering, where the first side of each member is common to the second side of the previous member —  
 mlib\_GraphicsDrawTriangleStripSet\_AZ\_32, 150

draw triangle set with antialiasing and Z buffering, where the first side of each member is common to the second side of the previous member —  
 mlib\_GraphicsDrawTriangleStripSet\_AZ\_8, 150

draw triangle set with antialiasing where all members of the set have a common vertex —  
 mlib\_GraphicsDrawTriangleFanSet\_A\_32, 124

draw triangle set with antialiasing where all members of the set have a common vertex —  
 mlib\_GraphicsDrawTriangleFanSet\_A\_8, 124

draw triangle set with antialiasing where each member can have different vertices —  
 mlib\_GraphicsDrawTriangleSet\_A\_32, 137

draw triangle set with antialiasing where each member can have different vertices —  
 mlib\_GraphicsDrawTriangleSet\_A\_8, 137

draw triangle set with antialiasing where the first side of each member is common to the second side of the previous member —  
 mlib\_GraphicsDrawTriangleStripSet\_A\_32, 146

draw triangle set with antialiasing where the first side of each member is common to the second side of the previous member —  
 mlib\_GraphicsDrawTriangleStripSet\_A\_8, 146

draw triangle set with antialiasing, Gouraud shading, and Z buffering, where all members of the set have a common vertex —  
 mlib\_GraphicsDrawTriangleFanSet\_AGZ\_32, 126

draw triangle set with antialiasing, Gouraud shading, and Z buffering, where all members of the set have a common vertex —  
 mlib\_GraphicsDrawTriangleFanSet\_AGZ\_8, 126

draw triangle set with antialiasing, Gouraud shading, and Z buffering, where each member can have different vertices —  
 mlib\_GraphicsDrawTriangleSet\_AGZ\_32, 139

draw triangle set with antialiasing, Gouraud shading, and Z buffering, where each member can have different vertices —  
 mlib\_GraphicsDrawTriangleSet\_AGZ\_8, 139

draw triangle set with antialiasing, Gouraud shading, and Z buffering, where the first side of each member is common to the second side of the previous member —  
 mlib\_GraphicsDrawTriangleStripSet\_AGZ, 148

draw triangle set with antialiasing, Gouraud shading, and Z buffering, where the first side of each member is common to the second side of the previous member —  
 mlib\_GraphicsDrawTriangleStripSet\_AGZ\_32, 148

draw triangle set with antialiasing, Gouraud shading, and Z buffering, where the first side of each member is common to the second side of the previous member —  
 mlib\_GraphicsDrawTriangleStripSet\_AGZ\_8, 148

draw triangle set with Gouraud shading and Z buffering, where all members of the set have a common vertex —  
 mlib\_GraphicsDrawTriangleFanSet\_GZ\_32, 129

draw triangle set with Gouraud shading and Z buffering, where all members of the set have a common vertex —  
 mlib\_GraphicsDrawTriangleFanSet\_GZ\_8, 129

draw triangle set with Gouraud shading and Z buffering, where each member can have different vertices —  
 mlib\_GraphicsDrawTriangleSet\_GZ\_32, 142

draw triangle set with Gouraud shading and Z buffering, where each member can have different vertices —  
 mlib\_GraphicsDrawTriangleSet\_GZ\_8, 142



- draw triangle set with Gouraud shading and Z buffering, where the first side of each member is common to the second side of the previous member —  
    - mllib\_GraphicsDrawTriangleStripSet\_GZ\_32, 152
  - draw triangle set with Gouraud shading and Z buffering, where the first side of each member is common to the second side of the previous member —  
    - mllib\_GraphicsDrawTriangleStripSet\_GZ\_8, 152
  - draw triangle set with Gouraud shading where all members of the set have a common vertex —  
    - mllib\_GraphicsDrawTriangleFanSet\_G\_32, 128
  - draw triangle set with Gouraud shading where all members of the set have a common vertex —  
    - mllib\_GraphicsDrawTriangleFanSet\_G\_8, 128
  - draw triangle set with Gouraud shading where each member can have different vertices —  
    - mllib\_GraphicsDrawTriangleSet\_G\_32, 141
  - draw triangle set with Gouraud shading where each member can have different vertices —  
    - mllib\_GraphicsDrawTriangleSet\_G\_8, 141
  - draw triangle set with Gouraud shading where the first side of each member is common to the second side of the previous member —  
    - mllib\_GraphicsDrawTriangleStripSet\_G\_32, 151
  - draw triangle set with Gouraud shading where the first side of each member is common to the second side of the previous member —  
    - mllib\_GraphicsDrawTriangleStripSet\_G\_8, 151
  - draw triangle set with Z buffering —  
    - mllib\_GraphicsDrawTriangleSet\_Z\_32, 144
  - draw triangle set with Z buffering —  
    - mllib\_GraphicsDrawTriangleSet\_Z\_8, 144
  - draw triangle set with Z buffering where all members of the set have a common vertex —  
    - mllib\_GraphicsDrawTriangleFanSet\_Z\_32, 131
  - draw triangle set with Z buffering where all members of the set have a common vertex —  
    - mllib\_GraphicsDrawTriangleFanSet\_Z\_8, 131
  - draw triangle set with Z buffering where the first side of each member is common to the second side of the previous member —  
    - mllib\_GraphicsDrawTriangleStripSet\_Z\_32, 154
  - draw triangle set with Z buffering where the first side of each member is common to the second side of the previous member —  
    - mllib\_GraphicsDrawTriangleStripSet\_Z\_8, 154
  - draw triangle with antialiasing —  
    - mllib\_GraphicsDrawTriangle\_A\_32, 116
  - draw triangle with antialiasing —  
    - mllib\_GraphicsDrawTriangle\_A\_8, 116
  - draw triangle with antialiasing and Gouraud shading —  
    - mllib\_GraphicsDrawTriangle\_AG\_32, 117
  - draw triangle with antialiasing and Gouraud shading —  
    - mllib\_GraphicsDrawTriangle\_AG\_8, 117
  - draw triangle with antialiasing and Z buffering —  
    - mllib\_GraphicsDrawTriangle\_AZ\_32, 121
  - draw triangle with antialiasing and Z buffering —  
    - mllib\_GraphicsDrawTriangle\_AZ\_8, 121
  - draw triangle with antialiasing, Gouraud shading, and Z buffering —  
    - mllib\_GraphicsDrawTriangle\_AGZ\_32, 119
  - draw triangle with antialiasing, Gouraud shading, and Z buffering —  
    - mllib\_GraphicsDrawTriangle\_AGZ\_8, 119
  - draw triangle with Gouraud shading —  
    - mllib\_GraphicsDrawTriangle\_G\_32, 132
  - draw triangle with Gouraud shading —  
    - mllib\_GraphicsDrawTriangle\_G\_8, 132
  - draw triangle with Gouraud shading and Z buffering —  
    - mllib\_GraphicsDrawTriangle\_GZ\_32, 134
  - draw triangle with Gouraud shading and Z buffering —  
    - mllib\_GraphicsDrawTriangle\_GZ\_8, 134
  - draw triangle with Z buffering —  
    - mllib\_GraphicsDrawTriangle\_Z\_32, 157
  - draw triangle with Z buffering —  
    - mllib\_GraphicsDrawTriangle\_Z\_8, 157
- E**
- eight neighbor dilate —  
    - mllib\_ImageDilate8, 429
  - eight neighbor dilate —  
    - mllib\_ImageDilate8\_Fp, 430
  - eight neighbor erode —  
    - mllib\_ImageErode8, 447

eight neighbor erode —  
mllib\_ImageErode8\_Fp, 448

## F

find the first element with the maximum magnitude in a matrix —  
mllib\_MatrixMaximumMag\_D64C, 803  
find the first element with the maximum magnitude in a matrix —  
mllib\_MatrixMaximumMag\_F32C, 803  
find the first element with the maximum magnitude in a matrix —  
mllib\_MatrixMaximumMag\_S16C, 803  
find the first element with the maximum magnitude in a matrix —  
mllib\_MatrixMaximumMag\_S32C, 803  
find the first element with the maximum magnitude in a matrix —  
mllib\_MatrixMaximumMag\_S8C, 803  
find the first element with the maximum magnitude in a matrix —  
mllib\_MatrixMaximumMag\_U8C, 803  
find the first element with the maximum magnitude in a vector —  
mllib\_VectorMaximumMag\_D64C, 1227  
find the first element with the maximum magnitude in a vector —  
mllib\_VectorMaximumMag\_F32C, 1227  
find the first element with the maximum magnitude in a vector —  
mllib\_VectorMaximumMag\_S16C, 1227  
find the first element with the maximum magnitude in a vector —  
mllib\_VectorMaximumMag\_S32C, 1227  
find the first element with the maximum magnitude in a vector —  
mllib\_VectorMaximumMag\_S8C, 1227  
find the first element with the maximum magnitude in a vector —  
mllib\_VectorMaximumMag\_U8C, 1227  
find the first element with the minimum magnitude in a matrix —  
mllib\_MatrixMinimumMag\_D64C, 805  
find the first element with the minimum magnitude in a matrix —  
mllib\_MatrixMinimumMag\_F32C, 805

find the first element with the minimum magnitude in a matrix —  
mllib\_MatrixMinimumMag\_S16C, 805  
find the first element with the minimum magnitude in a matrix —  
mllib\_MatrixMinimumMag\_S32C, 805  
find the first element with the minimum magnitude in a matrix —  
mllib\_MatrixMinimumMag\_S8C, 805  
find the first element with the minimum magnitude in a matrix —  
mllib\_MatrixMinimumMag\_U8C, 805  
find the first element with the minimum magnitude in a vector —  
mllib\_VectorMinimumMag\_D64C, 1230  
find the first element with the minimum magnitude in a vector —  
mllib\_VectorMinimumMag\_F32C, 1230  
find the first element with the minimum magnitude in a vector —  
mllib\_VectorMinimumMag\_S16C, 1230  
find the first element with the minimum magnitude in a vector —  
mllib\_VectorMinimumMag\_S32C, 1230  
find the first element with the minimum magnitude in a vector —  
mllib\_VectorMinimumMag\_S8C, 1230  
find the first element with the minimum magnitude in a vector —  
mllib\_VectorMinimumMag\_U8C, 1230  
find the maximum value in a matrix —  
mllib\_MatrixMaximum\_D64, 804  
find the maximum value in a matrix —  
mllib\_MatrixMaximum\_F32, 804  
find the maximum value in a matrix —  
mllib\_MatrixMaximum\_S16, 804  
find the maximum value in a matrix —  
mllib\_MatrixMaximum\_S32, 804  
find the maximum value in a matrix —  
mllib\_MatrixMaximum\_S8, 804  
find the maximum value in a matrix —  
mllib\_MatrixMaximum\_U8, 804  
find the maximum value in a vector —  
mllib\_VectorMaximum\_D64, 1228  
find the maximum value in a vector —  
mllib\_VectorMaximum\_F32, 1228  
find the maximum value in a vector —  
mllib\_VectorMaximum\_S16, 1228

- find the maximum value in a vector —  
  mlib\_VectorMaximum\_S32, 1228
- find the maximum value in a vector —  
  mlib\_VectorMaximum\_S8, 1228
- find the maximum value in a vector —  
  mlib\_VectorMaximum\_U8, 1228
- find the minimum value in a matrix —  
  mlib\_MatrixMinimum\_D64, 806
- find the minimum value in a matrix —  
  mlib\_MatrixMinimum\_F32, 806
- find the minimum value in a matrix —  
  mlib\_MatrixMinimum\_S16, 806
- find the minimum value in a matrix —  
  mlib\_MatrixMinimum\_S32, 806
- find the minimum value in a matrix —  
  mlib\_MatrixMinimum\_S8, 806
- find the minimum value in a matrix —  
  mlib\_MatrixMinimum\_U8, 806
- find the minimum value in a vector —  
  mlib\_VectorMinimum\_D64, 1231
- find the minimum value in a vector —  
  mlib\_VectorMinimum\_F32, 1231
- find the minimum value in a vector —  
  mlib\_VectorMinimum\_S16, 1231
- find the minimum value in a vector —  
  mlib\_VectorMinimum\_S32, 1231
- find the minimum value in a vector —  
  mlib\_VectorMinimum\_S8, 1231
- find the minimum value in a vector —  
  mlib\_VectorMinimum\_U8, 1231
- Finite Impulse Response (FIR) filtering —  
  mlib\_SignalFIRFree\_S16S\_S16S, 981
- Finite Impulse Response (FIR) filtering —  
  mlib\_SignalFIRFree\_S16\_S16, 981
- Finite Impulse Response (FIR) filtering —  
  mlib\_SignalFIRInit\_S16S\_S16S, 984
- Finite Impulse Response (FIR) filtering —  
  mlib\_SignalFIRInit\_S16\_S16, 984
- Finite Impulse Response (FIR) filtering —  
  mlib\_SignalFIR\_S16S\_S16S\_Sat, 985
- Finite Impulse Response (FIR) filtering —  
  mlib\_SignalFIR\_S16\_S16\_Sat, 985
- Finite Impulse Response (FIR) filtering —  
  mlib\_SignalFIRFree\_F32S\_F32S, 980
- Finite Impulse Response (FIR) filtering —  
  mlib\_SignalFIRFree\_F32\_F32, 979
- Finite Impulse Response (FIR) filtering —  
  mlib\_SignalFIRInit\_F32S\_F32S, 983
- Finite Impulse Response (FIR) filtering —  
  mlib\_SignalFIRInit\_F32\_F32, 982
- Finite Impulse Response (FIR) filtering —  
  mlib\_SignalFIR\_F32S\_F32S, 978
- Finite Impulse Response (FIR) filtering —  
  mlib\_SignalFIR\_F32\_F32, 977
- float to 16-bit quantization —  
  mlib\_SignalQuant2\_S16S\_F32S, 1159
- float to 16-bit quantization —  
  mlib\_SignalQuant2\_S16\_F32, 1158
- float to 16-bit quantization —  
  mlib\_SignalQuant\_S16S\_F32S, 1161
- float to 16-bit quantization —  
  mlib\_SignalQuant\_S16\_F32, 1160
- float to 8-bit quantization —  
  mlib\_SignalQuant\_U8S\_F32S, 1164
- float to 8-bit quantization —  
  mlib\_SignalQuant\_U8\_F32, 1162
- flood fill — mlib\_GraphicsFloodFill\_32, 222
- flood fill — mlib\_GraphicsFloodFill\_8, 222
- forward Discrete Cosine Transform (DCT) —  
  mlib\_VideoDCT16x16\_S16\_S16, 1443
- forward Discrete Cosine Transform (DCT) —  
  mlib\_VideoDCT16x16\_S16\_S16\_B10, 1444
- forward Discrete Cosine Transform (DCT) —  
  mlib\_VideoDCT2x2\_S16\_S16, 1445
- forward Discrete Cosine Transform (DCT) —  
  mlib\_VideoDCT4x4\_S16\_S16, 1446
- forward Discrete Cosine Transform (DCT) —  
  mlib\_VideoDCT8x8\_S16\_S16, 1447
- forward Discrete Cosine Transform (DCT) —  
  mlib\_VideoDCT8x8\_S16\_S16\_B12, 1448
- forward Discrete Cosine Transform (DCT) —  
  mlib\_VideoDCT8x8\_S16\_S16\_NA, 1449
- forward Discrete Cosine Transform (DCT) —  
  mlib\_VideoDCT8x8\_S16\_U8, 1450
- forward Discrete Cosine Transform (DCT) —  
  mlib\_VideoDCT8x8\_S16\_U8\_NA, 1451
- four neighbor dilate — mlib\_ImageDilate4, 427
- four neighbor dilate —  
  mlib\_ImageDilate4\_Fp, 428
- four neighbor erode — mlib\_ImageErode4, 445
- four neighbor erode —  
  mlib\_ImageErode4\_Fp, 446
- Fourier transform —  
  mlib\_ImageFourierTransform, 469
- free a block of bytes — mlib\_free, 40

## G

Gaussian noise generation —  
  mllib\_SignalGaussNoiseFree\_F32, 987

Gaussian noise generation —  
  mllib\_SignalGaussNoiseFree\_S16, 988

Gaussian noise generation —  
  mllib\_SignalGaussNoiseInit\_F32, 989

Gaussian noise generation —  
  mllib\_SignalGaussNoiseInit\_S16, 990

Gaussian noise generation —  
  mllib\_SignalGaussNoise\_F32, 986

Gaussian noise generation —  
  mllib\_SignalGaussNoise\_S16, 991

generates the 8x8 luminance prediction block in  
the Advanced Prediction Mode for H.263  
codec —  
  mllib\_VideoH263OverlappedMC\_S16\_U8, 1458

generates the 8x8 luminance prediction block in  
the Advanced Prediction Mode for H.263  
codec —  
  mllib\_VideoH263OverlappedMC\_U8\_U8, 1461

get bitoffset — mllib\_ImageGetBitOffset, 471

get channels — mllib\_ImageGetChannels, 472

get data — mllib\_ImageGetData, 473

get flags — mllib\_ImageGetFlags, 474

get format — mllib\_ImageGetFormat, 475

get height — mllib\_ImageGetHeight, 476

get paddings — mllib\_ImageGetPaddings, 477

get stride — mllib\_ImageGetStride, 478

get type — mllib\_ImageGetType, 479

get width — mllib\_ImageGetWidth, 480

grid-based image warp —  
  mllib\_ImageGridWarp, 489

grid-based image warp —  
  mllib\_ImageGridWarp\_Fp, 492

grid-based image warp with table-driven  
interpolation —  
  mllib\_ImageGridWarpTable, 495

grid-based image warp with table-driven  
interpolation —  
  mllib\_ImageGridWarpTable\_Fp, 498

## H

half-pixel interpolation in the X and Y directions  
— mllib\_VideoInterpXY\_U8\_U8\_16x16, 1498

half-pixel interpolation in the X and Y directions  
— mllib\_VideoInterpXY\_U8\_U8\_16x8, 1498

half-pixel interpolation in the X and Y directions  
— mllib\_VideoInterpXY\_U8\_U8\_8x16, 1498

half-pixel interpolation in the X and Y directions  
— mllib\_VideoInterpXY\_U8\_U8\_8x4, 1498

half-pixel interpolation in the X and Y directions  
— mllib\_VideoInterpXY\_U8\_U8\_8x8, 1498

half-pixel interpolation in the X and Y directions  
and averaging for reference block —  
  mllib\_VideoInterpAveXY\_U8\_U8\_16x16, 1478

half-pixel interpolation in the X and Y directions  
and averaging for reference block —  
  mllib\_VideoInterpAveXY\_U8\_U8\_16x8, 1478

half-pixel interpolation in the X and Y directions  
and averaging for reference block —  
  mllib\_VideoInterpAveXY\_U8\_U8\_8x16, 1478

half-pixel interpolation in the X and Y directions  
and averaging for reference block —  
  mllib\_VideoInterpAveXY\_U8\_U8\_8x4, 1478

half-pixel interpolation in the X and Y directions  
and averaging for reference block —  
  mllib\_VideoInterpAveXY\_U8\_U8\_8x8, 1478

half-pixel interpolation in the X and Y directions  
for motion compensation —  
  mllib\_VideoInterpXY\_S16\_U8\_16x16, 1494

half-pixel interpolation in the X and Y directions  
for motion compensation —  
  mllib\_VideoInterpXY\_S16\_U8\_16x8, 1494

half-pixel interpolation in the X and Y directions  
for motion compensation —  
  mllib\_VideoInterpXY\_S16\_U8\_8x16, 1494

half-pixel interpolation in the X and Y directions  
for motion compensation —  
  mllib\_VideoInterpXY\_S16\_U8\_8x4, 1494

half-pixel interpolation in the X and Y directions  
for motion compensation —  
  mllib\_VideoInterpXY\_S16\_U8\_8x8, 1494

half-pixel interpolation in the X direction —  
  mllib\_VideoInterpX\_S16\_U8\_16x16, 1486

half-pixel interpolation in the X direction —  
  mllib\_VideoInterpX\_S16\_U8\_16x8, 1486

half-pixel interpolation in the X direction —  
  mllib\_VideoInterpX\_S16\_U8\_8x16, 1486

half-pixel interpolation in the X direction —  
  mllib\_VideoInterpX\_S16\_U8\_8x4, 1486

half-pixel interpolation in the X direction —  
  mllib\_VideoInterpX\_S16\_U8\_8x8, 1486

half-pixel interpolation in the X direction —  
     mllib\_VideoInterpX\_U8\_U8\_16x16, 1490  
 half-pixel interpolation in the X direction —  
     mllib\_VideoInterpX\_U8\_U8\_16x8, 1490  
 half-pixel interpolation in the X direction —  
     mllib\_VideoInterpX\_U8\_U8\_8x16, 1490  
 half-pixel interpolation in the X direction —  
     mllib\_VideoInterpX\_U8\_U8\_8x4, 1490  
 half-pixel interpolation in the X direction —  
     mllib\_VideoInterpX\_U8\_U8\_8x8, 1490  
 half-pixel interpolation in the X direction and  
 averaging for reference block —  
     mllib\_VideoInterpAveX\_U8\_U8\_16x16, 1474  
 half-pixel interpolation in the X direction and  
 averaging for reference block —  
     mllib\_VideoInterpAveX\_U8\_U8\_16x8, 1474  
 half-pixel interpolation in the X direction and  
 averaging for reference block —  
     mllib\_VideoInterpAveX\_U8\_U8\_8x16, 1474  
 half-pixel interpolation in the X direction and  
 averaging for reference block —  
     mllib\_VideoInterpAveX\_U8\_U8\_8x4, 1474  
 half-pixel interpolation in the X direction and  
 averaging for reference block —  
     mllib\_VideoInterpAveX\_U8\_U8\_8x8, 1474  
 half-pixel interpolation in the Y direction —  
     mllib\_VideoInterpY\_S16\_U8\_16x16, 1503  
 half-pixel interpolation in the Y direction —  
     mllib\_VideoInterpY\_S16\_U8\_16x8, 1503  
 half-pixel interpolation in the Y direction —  
     mllib\_VideoInterpY\_S16\_U8\_8x16, 1503  
 half-pixel interpolation in the Y direction —  
     mllib\_VideoInterpY\_S16\_U8\_8x4, 1503  
 half-pixel interpolation in the Y direction —  
     mllib\_VideoInterpY\_S16\_U8\_8x8, 1503  
 half-pixel interpolation in the Y direction —  
     mllib\_VideoInterpY\_U8\_U8\_16x16, 1507  
 half-pixel interpolation in the Y direction —  
     mllib\_VideoInterpY\_U8\_U8\_16x8, 1507  
 half-pixel interpolation in the Y direction —  
     mllib\_VideoInterpY\_U8\_U8\_8x16, 1507  
 half-pixel interpolation in the Y direction —  
     mllib\_VideoInterpY\_U8\_U8\_8x4, 1507  
 half-pixel interpolation in the Y direction —  
     mllib\_VideoInterpY\_U8\_U8\_8x8, 1507  
 half-pixel interpolation in the Y direction and  
 averaging for reference block —  
     mllib\_VideoInterpAveY\_U8\_U8\_16x16, 1482  
 half-pixel interpolation in the Y direction and  
 averaging for reference block —  
     mllib\_VideoInterpAveY\_U8\_U8\_16x8, 1482  
 half-pixel interpolation in the Y direction and  
 averaging for reference block —  
     mllib\_VideoInterpAveY\_U8\_U8\_8x16, 1482  
 half-pixel interpolation in the Y direction and  
 averaging for reference block —  
     mllib\_VideoInterpAveY\_U8\_U8\_8x4, 1482  
 half-pixel interpolation in the Y direction and  
 averaging for reference block —  
     mllib\_VideoInterpAveY\_U8\_U8\_8x8, 1482  
 half-pixel interpolation in both X and Y  
 directions for replenishment mode —  
     mllib\_VideoInterpX\_Y\_XY\_U8\_U8, 1502  
 half-pixel interpolation in the X and Y directions  
 — mllib\_VideoInterpXY\_S16\_U8, 1496  
 half-pixel interpolation in the X and Y directions  
 — mllib\_VideoInterpXY\_U8\_U8, 1500  
 half-pixel interpolation in the X and Y directions  
 and averaging for reference block —  
     mllib\_VideoInterpAveXY\_U8\_U8, 1480  
 half-pixel interpolation in the X direction —  
     mllib\_VideoInterpX\_S16\_U8, 1488  
 half-pixel interpolation in the X direction —  
     mllib\_VideoInterpX\_U8\_U8, 1492  
 half-pixel interpolation in the X direction and  
 averaging for reference block —  
     mllib\_VideoInterpAveX\_U8\_U8, 1476  
 half-pixel interpolation in the Y direction —  
     mllib\_VideoInterpY\_S16\_U8, 1505  
 half-pixel interpolation in the Y direction —  
     mllib\_VideoInterpY\_U8\_U8, 1509  
 half-pixel interpolation in the Y direction and  
 averaging for reference block —  
     mllib\_VideoInterpAveY\_U8\_U8, 1484  
 Hamming windowing multiplication —  
     mllib\_SignalMulHamming\_F32S\_F32S, 1121  
 Hamming windowing multiplication —  
     mllib\_SignalMulHamming\_F32\_F32, 1121  
 Hamming windowing multiplication —  
     mllib\_SignalMulHamming\_S16S\_S16S, 1123  
 Hamming windowing multiplication —  
     mllib\_SignalMulHamming\_S16\_S16, 1123  
 Hamming window generation —  
     mllib\_SignalGenHamming\_F32, 996  
 Hamming window generation —  
     mllib\_SignalGenHamming\_S16, 997

- Hanning windowing multiplication —  
  - mllib\_SignalMulHanning\_F32, 1124
- Hanning windowing multiplication —  
  - mllib\_SignalMulHanning\_F32S, 1124
- Hanning windowing multiplication —  
  - mllib\_SignalMulHanning\_F32S\_F32S, 1125
- Hanning windowing multiplication —  
  - mllib\_SignalMulHanning\_F32\_F32, 1125
- Hanning windowing multiplication —  
  - mllib\_SignalMulHanning\_S16, 1126
- Hanning windowing multiplication —  
  - mllib\_SignalMulHanning\_S16S, 1126
- Hanning windowing multiplication —  
  - mllib\_SignalMulHanning\_S16S\_S16S, 1127
- Hanning windowing multiplication —  
  - mllib\_SignalMulHanning\_S16\_S16, 1127
- Hanning window generation —  
  - mllib\_SignalGenHanning\_F32, 998
- Hanning window generation —  
  - mllib\_SignalGenHanning\_S16, 999
- histogram — mllib\_ImageHistogram2, 501
- histogram — mllib\_ImageHistogram, 503
- HSL to RGB color conversion —  
  - mllib\_ImageColorHSL2RGB, 300
- HSL to RGB color conversion —  
  - mllib\_ImageColorHSL2RGB\_Fp, 302
- HSV to RGB color conversion —  
  - mllib\_ImageColorHSV2RGB, 303
- HSV to RGB color conversion —  
  - mllib\_ImageColorHSV2RGB\_Fp, 305

**I**

- IEEE-1180 compliant inverse Discrete Cosine Transform —  
  - mllib\_VideoIDCT\_IEEE\_S16\_S16, 1473
- image blend —  
  - mllib\_VideoColorBlendABGR, 1303
- image blend —  
  - mllib\_VideoColorBlendABGR\_ResetAlpha, 1303
- image extrema — mllib\_ImageExtrema2, 453
- image extrema —  
  - mllib\_ImageExtrema2\_Fp, 453
- image extrema and their locations —  
  - mllib\_ImageExtremaLocations, 455
- image extrema and their locations —  
  - mllib\_ImageExtremaLocations\_Fp, 455
- image affine transformation —  
  - mllib\_ImageAffine, 231
- image affine transformation —  
  - mllib\_ImageAffine\_Fp, 233
- image blending and channel reordering —  
  - mllib\_ImageBlendRGBA2ARGB, 278
- image blending and channel reordering —  
  - mllib\_ImageBlendRGBA2BGRA, 279
- image blending with scalar —  
  - mllib\_ImageScalarBlend, 675
- image blending with scalar —  
  - mllib\_ImageScalarBlend\_Fp, 676
- image blending with scalar —  
  - mllib\_ImageScalarBlend\_Fp\_Inp, 677
- image blending with scalar, in place —  
  - mllib\_ImageScalarBlend\_Inp, 678
- image composition —  
  - mllib\_ImageComposite, 331
- image composition, in place —  
  - mllib\_ImageComposite\_Inp, 333
- image copy — mllib\_ImageCopy, 413
- image creation — mllib\_ImageCreate, 418
- image data buffer reformat —  
  - mllib\_ImageReformat, 656
- image delete — mllib\_ImageDelete, 426
- image maximum — mllib\_ImageMaximum, 550
- image maximum —  
  - mllib\_ImageMaximum\_Fp, 551
- image mean — mllib\_ImageMean, 553
- image mean — mllib\_ImageMean\_Fp, 554
- image minimum — mllib\_ImageMinimum, 594
- image minimum —  
  - mllib\_ImageMinimum\_Fp, 595
- image query, 1D vector —  
  - mllib\_ImageIsNotOneDvector, 518
- image query, 2X height —  
  - mllib\_ImageIsNotHeight2X, 515
- image query, 2X width —  
  - mllib\_ImageIsNotWidth2X, 520
- image query, 4X height —  
  - mllib\_ImageIsNotHeight4X, 516
- image query, 4X width —  
  - mllib\_ImageIsNotWidth4X, 521
- image query, 64-byte aligned —  
  - mllib\_ImageIsNotAligned64, 513
- image query, 8X height —  
  - mllib\_ImageIsNotHeight8X, 517

- image query, 8X stride —
  - mllib\_ImageIsNotStride8X, 519
- image query, 8X width —
  - mllib\_ImageIsNotWidth8X, 522
- image query, eight-byte aligned —
  - mllib\_ImageIsNotAligned8, 514
- image query, four-byte aligned —
  - mllib\_ImageIsNotAligned4, 512
- image query, two-byte aligned —
  - mllib\_ImageIsNotAligned2, 511
- image query, user-allocated —
  - mllib\_ImageIsUserAllocated, 523
- image resize —
  - mllib\_VideoColorResizeABGR, 1324
- image scaling using interpolation table,
  - combined with alpha blending —
    - mllib\_ImageZoomTranslateTableBlend, 784
- image scaling with alpha blending —
  - mllib\_ImageZoomBlend, 759
- image scaling with alpha blending —
  - mllib\_ImageZoomTranslateBlend, 777
- image standard deviation —
  - mllib\_ImageStdDev, 710
- image standard deviation —
  - mllib\_ImageStdDev\_Fp, 711
- image structure creation —
  - mllib\_ImageCreateStruct, 419
- image thresholding — mllib\_ImageThresh1, 723
- image thresholding —
  - mllib\_ImageThresh1\_Fp, 725
- image thresholding —
  - mllib\_ImageThresh1\_Fp\_Inp, 727
- image thresholding —
  - mllib\_ImageThresh1\_Inp, 728
- image thresholding — mllib\_ImageThresh2, 729
- image thresholding —
  - mllib\_ImageThresh2\_Fp, 730
- image thresholding —
  - mllib\_ImageThresh2\_Fp\_Inp, 731
- image thresholding —
  - mllib\_ImageThresh2\_Inp, 732
- image thresholding — mllib\_ImageThresh3, 733
- image thresholding —
  - mllib\_ImageThresh3\_Fp, 734
- image thresholding —
  - mllib\_ImageThresh3\_Fp\_Inp, 735
- image thresholding —
  - mllib\_ImageThresh3\_Inp, 736
- image thresholding — mllib\_ImageThresh4, 737
- image thresholding —
  - mllib\_ImageThresh4\_Fp, 739
- image thresholding —
  - mllib\_ImageThresh4\_Fp\_Inp, 741
- image thresholding —
  - mllib\_ImageThresh4\_Inp, 743
- image thresholding — mllib\_ImageThresh5, 745
- image thresholding —
  - mllib\_ImageThresh5\_Fp, 747
- image thresholding —
  - mllib\_ImageThresh5\_Fp\_Inp, 749
- image thresholding —
  - mllib\_ImageThresh5\_Inp, 750
- image X projection — mllib\_ImageXProj, 753
- image X projection —
  - mllib\_ImageXProj\_Fp, 754
- image Y projection — mllib\_ImageYProj, 755
- image Y projection —
  - mllib\_ImageYProj\_Fp, 756
- in-place image blend —
  - mllib\_VideoColorBlendABGR\_Inp, 1305
- in-place image blend —
  - mllib\_VideoColorBlendABGR\_ResetAlpha\_Inp, 1305
- initialization for autocorrelation method of
  - linear predictive coding —
    - mllib\_SignalLPCAutoCorrelInit\_F32, 1077
- initialization for autocorrelation method of
  - linear predictive coding —
    - mllib\_SignalLPCAutoCorrelInit\_S16, 1077
- initialization for cepstral analysis —
  - mllib\_SignalCepstralInit\_F32, 862
- initialization for cepstral analysis —
  - mllib\_SignalCepstralInit\_S16, 862
- initialization for cepstral analysis in mel
  - frequency scale —
    - mllib\_SignalMelCepstralInit\_F32, 1102
- initialization for cepstral analysis in mel
  - frequency scale —
    - mllib\_SignalMelCepstralInit\_S16, 1102
- initialization for covariance method of linear
  - predictive coding —
    - mllib\_SignalLPCCovarianceInit\_F32, 1083
- initialization for covariance method of linear
  - predictive coding —
    - mllib\_SignalLPCCovarianceInit\_S16, 1083

initialization for perceptual weighting of linear predictive coding —  
     mllib\_SignalLPCPerceptWeightInit\_F32, 1088  
 initialization for perceptual weighting of linear predictive coding —  
     mllib\_SignalLPCPerceptWeightInit\_S16, 1088  
 initialization for resampling with filtering —  
     mllib\_SignalReSampleFIRInit\_F32S\_F32S, 1170  
 initialization for resampling with filtering —  
     mllib\_SignalReSampleFIRInit\_F32\_F32, 1170  
 initialization for resampling with filtering —  
     mllib\_SignalReSampleFIRInit\_S16S\_S16S, 1170  
 initialization for resampling with filtering —  
     mllib\_SignalReSampleFIRInit\_S16\_S16, 1170  
 initialization for signal pre-emphasizing —  
     mllib\_SignalEmphasizeInit\_F32S\_F32S, 951  
 initialization for signal pre-emphasizing —  
     mllib\_SignalEmphasizeInit\_F32\_F32, 951  
 initialization for signal pre-emphasizing —  
     mllib\_SignalEmphasizeInit\_S16S\_S16S, 951  
 initialization for signal pre-emphasizing —  
     mllib\_SignalEmphasizeInit\_S16\_S16, 951  
 initialization for image dithering —  
     mllib\_ImageColorDitherInit, 294  
 initialization for K-best paths of scalar data —  
     mllib\_SignalDTWKScalarInit\_F32, 884  
 initialization for K-best paths of scalar data —  
     mllib\_SignalDTWKScalarInit\_S16, 885  
 initialization for K-best paths of vector data —  
     mllib\_SignalDTWKVectorInit\_F32, 899  
 initialization for K-best paths of vector data —  
     mllib\_SignalDTWKVectorInit\_S16, 901  
 initialization for scalar data —  
     mllib\_SignalDTWScalarInit\_F32, 916  
 initialization for scalar data —  
     mllib\_SignalDTWScalarInit\_S16, 917  
 initialization for true color to indexed color conversion —  
     mllib\_ImageColorTrue2IndexInit, 325  
 initialization for vector data —  
     mllib\_SignalDTWVectorInit\_F32, 935  
 initialization for vector data —  
     mllib\_SignalDTWVectorInit\_S16, 936  
 initialize vector to zero —  
     mllib\_VectorZero\_S16, 1287  
 initialize vector to zero —  
     mllib\_VectorZero\_S16C, 1287  
 initialize vector to zero —  
     mllib\_VectorZero\_S32, 1287  
 initialize vector to zero —  
     mllib\_VectorZero\_S32C, 1287  
 initialize vector to zero —  
     mllib\_VectorZero\_S8, 1287  
 initialize vector to zero —  
     mllib\_VectorZero\_S8C, 1287  
 initialize vector to zero —  
     mllib\_VectorZero\_U8, 1287  
 initialize vector to zero —  
     mllib\_VectorZero\_U8C, 1287  
 inverse Discrete Cosine Transform —  
     mllib\_VideoIDCT8x8\_S16\_S16, 1464  
 inverse Discrete Cosine Transform —  
     mllib\_VideoIDCT8x8\_S16\_S16\_DC, 1465  
 inverse Discrete Cosine Transform —  
     mllib\_VideoIDCT8x8\_S16\_S16\_NA, 1466  
 inverse Discrete Cosine Transform —  
     mllib\_VideoIDCT8x8\_S16\_S16\_Q1, 1467  
 inverse Discrete Cosine Transform —  
     mllib\_VideoIDCT8x8\_S16\_S16\_Q1\_Mismatch, 1468  
 inverse Discrete Cosine Transform —  
     mllib\_VideoIDCT8x8\_U8\_S16, 1469  
 inverse Discrete Cosine Transform —  
     mllib\_VideoIDCT8x8\_U8\_S16\_DC, 1470  
 inverse Discrete Cosine Transform —  
     mllib\_VideoIDCT8x8\_U8\_S16\_NA, 1471  
 inverse Discrete Cosine Transform —  
     mllib\_VideoIDCT8x8\_U8\_S16\_Q1, 1472  
 invert — mllib\_ImageInvert, 507  
 invert — mllib\_ImageInvert\_Fp, 508  
 invert — mllib\_ImageInvert\_Fp\_Inp, 509  
 invert in place — mllib\_ImageInvert\_Inp, 510  
 ITU G.711 m-law and A-law compression and decompression —  
     mllib\_SignalALaw2Linear, 855  
 ITU G.711 m-law and A-law compression and decompression —  
     mllib\_SignalALaw2uLaw, 856  
 ITU G.711 m-law and A-law compression and decompression —  
     mllib\_SignalLinear2ALaw, 1048  
 ITU G.711 m-law and A-law compression and decompression —  
     mllib\_SignalLinear2uLaw, 1049



ITU G.711 m-law and A-law compression and  
decompression —  
mllib\_SignaluLaw2ALaw, 1181  
ITU G.711 m-law and A-law compression and  
decompression —  
mllib\_SignaluLaw2Linear, 1182

## J

JFIF YCbCr to ABGR color conversion —  
mllib\_VideoColorJFIFYCC2ABGR444, 1308  
JFIF YCbCr to ARGB color conversion —  
mllib\_VideoColorJFIFYCC2ARGB444, 1309  
JFIF YCbCr to CMYK color conversion —  
mllib\_VideoColorJFIFYCCK2CMYK444, 1317  
JFIF YCbCr to RGB color conversion —  
mllib\_VideoColorJFIFYCC2RGB420, 1310  
JFIF YCbCr to RGB color conversion —  
mllib\_VideoColorJFIFYCC2RGB420\_Nearest, 1312  
JFIF YCbCr to RGB color conversion —  
mllib\_VideoColorJFIFYCC2RGB422, 1313  
JFIF YCbCr to RGB color conversion —  
mllib\_VideoColorJFIFYCC2RGB422\_Nearest, 1314  
JFIF YCbCr to RGB color conversion —  
mllib\_VideoColorJFIFYCC2RGB444, 1315  
JFIF YCbCr to RGB color conversion —  
mllib\_VideoColorJFIFYCC2RGB444\_S16, 1316

## K

Kaiser windowing multiplication —  
mllib\_SignalMulKaiser\_F32, 1128  
Kaiser windowing multiplication —  
mllib\_SignalMulKaiser\_F32S, 1128  
Kaiser windowing multiplication —  
mllib\_SignalMulKaiser\_F32S\_F32S, 1129  
Kaiser windowing multiplication —  
mllib\_SignalMulKaiser\_F32\_F32, 1129  
Kaiser windowing multiplication —  
mllib\_SignalMulKaiser\_S16, 1130  
Kaiser windowing multiplication —  
mllib\_SignalMulKaiser\_S16S, 1130  
Kaiser windowing multiplication —  
mllib\_SignalMulKaiser\_S16S\_S16S, 1131  
Kaiser windowing multiplication —  
mllib\_SignalMulKaiser\_S16\_S16, 1131

Kaiser window generation —  
mllib\_SignalGenKaiser\_F32, 1000  
Kaiser window generation —  
mllib\_SignalGenKaiser\_S16, 1001  
kernel conversion for separable convolution —  
mllib\_ImageSConvKernelConvert, 700

## L

least mean square (LMS) adaptive filtering —  
mllib\_SignalLMSFilterFree\_F32S\_F32S, 1051  
least mean square (LMS) adaptive filtering —  
mllib\_SignalLMSFilterFree\_F32\_F32, 1051  
least mean square (LMS) adaptive filtering —  
mllib\_SignalLMSFilterFree\_S16S\_S16S, 1052  
least mean square (LMS) adaptive filtering —  
mllib\_SignalLMSFilterFree\_S16\_S16, 1052  
least mean square (LMS) adaptive filtering —  
mllib\_SignalLMSFilterInit\_F32S\_F32S, 1053  
least mean square (LMS) adaptive filtering —  
mllib\_SignalLMSFilterInit\_F32\_F32, 1053  
least mean square (LMS) adaptive filtering —  
mllib\_SignalLMSFilterInit\_S16S\_S16S, 1054  
least mean square (LMS) adaptive filtering —  
mllib\_SignalLMSFilterInit\_S16\_S16, 1054  
least mean square (LMS) adaptive filtering —  
mllib\_SignalLMSFilter\_F32S\_F32S, 1050  
least mean square (LMS) adaptive filtering —  
mllib\_SignalLMSFilter\_F32\_F32, 1050  
least mean square (LMS) adaptive filtering —  
mllib\_SignalLMSFilter\_S16S\_S16S\_Sat, 1055  
least mean square (LMS) adaptive filtering —  
mllib\_SignalLMSFilter\_S16\_S16\_Sat, 1055  
linear scaling — mllib\_ImageScale2, 679  
linear scaling — mllib\_ImageScale, 682  
linear scaling — mllib\_ImageScale\_Fp, 684  
linear scaling, in place —  
mllib\_ImageScale2\_Inp, 681  
linear scaling, in place —  
mllib\_ImageScale\_Fp\_Inp, 686  
linear scaling, in place —  
mllib\_ImageScale\_Inp, 687

## M

- main diagonal flip —
  - mllib\_ImageFlipMainDiag, 463
- main diagonal flip —
  - mllib\_ImageFlipMainDiag\_Fp, 464
- matrix addition —
  - mllib\_MatrixAdd\_S16C\_S16C\_Mod, 800
- matrix addition —
  - mllib\_MatrixAdd\_S16C\_S16C\_Sat, 800
- matrix addition —
  - mllib\_MatrixAdd\_S16C\_S8C\_Mod, 800
- matrix addition —
  - mllib\_MatrixAdd\_S16C\_S8C\_Sat, 800
- matrix addition —
  - mllib\_MatrixAdd\_S16C\_U8C\_Mod, 800
- matrix addition —
  - mllib\_MatrixAdd\_S16C\_U8C\_Sat, 800
- matrix addition —
  - mllib\_MatrixAdd\_S16\_S16\_Mod, 800
- matrix addition —
  - mllib\_MatrixAdd\_S16\_S16\_Sat, 800
- matrix addition —
  - mllib\_MatrixAdd\_S16\_S8\_Mod, 800
- matrix addition —
  - mllib\_MatrixAdd\_S16\_S8\_Sat, 800
- matrix addition —
  - mllib\_MatrixAdd\_S16\_U8\_Mod, 800
- matrix addition —
  - mllib\_MatrixAdd\_S16\_U8\_Sat, 800
- matrix addition —
  - mllib\_MatrixAdd\_S32C\_S16C\_Mod, 800
- matrix addition —
  - mllib\_MatrixAdd\_S32C\_S16C\_Sat, 800
- matrix addition —
  - mllib\_MatrixAdd\_S32C\_S32C\_Mod, 800
- matrix addition —
  - mllib\_MatrixAdd\_S32C\_S32C\_Sat, 800
- matrix addition —
  - mllib\_MatrixAdd\_S32\_S16\_Mod, 800
- matrix addition —
  - mllib\_MatrixAdd\_S32\_S16\_Sat, 800
- matrix addition —
  - mllib\_MatrixAdd\_S32\_S32\_Mod, 800
- matrix addition —
  - mllib\_MatrixAdd\_S32\_S32\_Sat, 800
- matrix addition —
  - mllib\_MatrixAdd\_S8C\_S8C\_Mod, 800
- matrix addition —
  - mllib\_MatrixAdd\_S8C\_S8C\_Sat, 800
- matrix addition —
  - mllib\_MatrixAdd\_S8\_S8\_Mod, 800
- matrix addition —
  - mllib\_MatrixAdd\_S8\_S8\_Sat, 800
- matrix addition —
  - mllib\_MatrixAdd\_U8C\_U8C\_Mod, 800
- matrix addition —
  - mllib\_MatrixAdd\_U8C\_U8C\_Sat, 800
- matrix addition —
  - mllib\_MatrixAdd\_U8\_U8\_Mod, 800
- matrix addition —
  - mllib\_MatrixAdd\_U8\_U8\_Sat, 800
- matrix addition to scalar —
  - mllib\_MatrixAddS\_S16C\_S16C\_Mod, 795
- matrix addition to scalar —
  - mllib\_MatrixAddS\_S16C\_S16C\_Sat, 795
- matrix addition to scalar —
  - mllib\_MatrixAddS\_S16C\_S8C\_Mod, 795
- matrix addition to scalar —
  - mllib\_MatrixAddS\_S16C\_S8C\_Sat, 795
- matrix addition to scalar —
  - mllib\_MatrixAddS\_S16C\_U8C\_Mod, 795
- matrix addition to scalar —
  - mllib\_MatrixAddS\_S16C\_U8C\_Sat, 795
- matrix addition to scalar —
  - mllib\_MatrixAddS\_S16\_S16\_Mod, 795
- matrix addition to scalar —
  - mllib\_MatrixAddS\_S16\_S16\_Sat, 795
- matrix addition to scalar —
  - mllib\_MatrixAddS\_S16\_S8\_Mod, 795
- matrix addition to scalar —
  - mllib\_MatrixAddS\_S16\_S8\_Sat, 795
- matrix addition to scalar —
  - mllib\_MatrixAddS\_S16\_U8\_Mod, 795
- matrix addition to scalar —
  - mllib\_MatrixAddS\_S16\_U8\_Sat, 795
- matrix addition to scalar —
  - mllib\_MatrixAddS\_S32C\_S16C\_Mod, 795
- matrix addition to scalar —
  - mllib\_MatrixAddS\_S32C\_S16C\_Sat, 795
- matrix addition to scalar —
  - mllib\_MatrixAddS\_S32C\_S32C\_Mod, 795
- matrix addition to scalar —
  - mllib\_MatrixAddS\_S32C\_S32C\_Sat, 795
- matrix addition to scalar —
  - mllib\_MatrixAddS\_S32\_S16\_Mod, 795

matrix addition to scalar —  
     mllib\_MatrixAddS\_S32\_S16\_Sat, 795  
 matrix addition to scalar —  
     mllib\_MatrixAddS\_S32\_S32\_Mod, 795  
 matrix addition to scalar —  
     mllib\_MatrixAddS\_S32\_S32\_Sat, 795  
 matrix addition to scalar —  
     mllib\_MatrixAddS\_S8C\_S8C\_Mod, 795  
 matrix addition to scalar —  
     mllib\_MatrixAddS\_S8C\_S8C\_Sat, 795  
 matrix addition to scalar —  
     mllib\_MatrixAddS\_S8\_S8\_Mod, 795  
 matrix addition to scalar —  
     mllib\_MatrixAddS\_S8\_S8\_Sat, 795  
 matrix addition to scalar —  
     mllib\_MatrixAddS\_U8C\_U8C\_Mod, 795  
 matrix addition to scalar —  
     mllib\_MatrixAddS\_U8C\_U8C\_Sat, 795  
 matrix addition to scalar —  
     mllib\_MatrixAddS\_U8\_U8\_Mod, 795  
 matrix addition to scalar —  
     mllib\_MatrixAddS\_U8\_U8\_Sat, 795  
 matrix addition to scalar, in place —  
     mllib\_MatrixAddS\_S16C\_Mod, 793  
 matrix addition to scalar, in place —  
     mllib\_MatrixAddS\_S16C\_Sat, 793  
 matrix addition to scalar, in place —  
     mllib\_MatrixAddS\_S16\_Mod, 793  
 matrix addition to scalar, in place —  
     mllib\_MatrixAddS\_S16\_Sat, 793  
 matrix addition to scalar, in place —  
     mllib\_MatrixAddS\_S32C\_Mod, 793  
 matrix addition to scalar, in place —  
     mllib\_MatrixAddS\_S32C\_Sat, 793  
 matrix addition to scalar, in place —  
     mllib\_MatrixAddS\_S32\_Mod, 793  
 matrix addition to scalar, in place —  
     mllib\_MatrixAddS\_S32\_Sat, 793  
 matrix addition to scalar, in place —  
     mllib\_MatrixAddS\_S8C\_Mod, 793  
 matrix addition to scalar, in place —  
     mllib\_MatrixAddS\_S8C\_Sat, 793  
 matrix addition to scalar, in place —  
     mllib\_MatrixAddS\_S8\_Mod, 793  
 matrix addition to scalar, in place —  
     mllib\_MatrixAddS\_S8\_Sat, 793  
 matrix addition to scalar, in place —  
     mllib\_MatrixAddS\_U8C\_Mod, 793  
 matrix addition to scalar, in place —  
     mllib\_MatrixAddS\_U8C\_Sat, 793  
 matrix addition to scalar, in place —  
     mllib\_MatrixAddS\_U8\_Mod, 793  
 matrix addition to scalar, in place —  
     mllib\_MatrixAddS\_U8\_Sat, 793  
 matrix addition, in place —  
     mllib\_MatrixAdd\_S16C\_Mod, 798  
 matrix addition, in place —  
     mllib\_MatrixAdd\_S16C\_Sat, 798  
 matrix addition, in place —  
     mllib\_MatrixAdd\_S16\_Mod, 798  
 matrix addition, in place —  
     mllib\_MatrixAdd\_S16\_Sat, 798  
 matrix addition, in place —  
     mllib\_MatrixAdd\_S32C\_Mod, 798  
 matrix addition, in place —  
     mllib\_MatrixAdd\_S32C\_Sat, 798  
 matrix addition, in place —  
     mllib\_MatrixAdd\_S32\_Mod, 798  
 matrix addition, in place —  
     mllib\_MatrixAdd\_S32\_Sat, 798  
 matrix addition, in place —  
     mllib\_MatrixAdd\_S8C\_Mod, 798  
 matrix addition, in place —  
     mllib\_MatrixAdd\_S8C\_Sat, 798  
 matrix addition, in place —  
     mllib\_MatrixAdd\_S8\_Mod, 798  
 matrix addition, in place —  
     mllib\_MatrixAdd\_S8\_Sat, 798  
 matrix addition, in place —  
     mllib\_MatrixAdd\_U8C\_Mod, 798  
 matrix addition, in place —  
     mllib\_MatrixAdd\_U8C\_Sat, 798  
 matrix addition, in place —  
     mllib\_MatrixAdd\_U8\_Mod, 798  
 matrix addition, in place —  
     mllib\_MatrixAdd\_U8\_Sat, 798  
 matrix linear scaling —  
     mllib\_MatrixScale\_S16C\_S16C\_Mod, 825  
 matrix linear scaling —  
     mllib\_MatrixScale\_S16C\_S16C\_Sat, 825  
 matrix linear scaling —  
     mllib\_MatrixScale\_S16C\_S8C\_Mod, 825  
 matrix linear scaling —  
     mllib\_MatrixScale\_S16C\_S8C\_Sat, 825  
 matrix linear scaling —  
     mllib\_MatrixScale\_S16C\_U8C\_Mod, 825

matrix linear scaling —  
     mllib\_MatrixScale\_S16C\_U8C\_Sat, 825  
 matrix linear scaling —  
     mllib\_MatrixScale\_S16\_S16\_Mod, 825  
 matrix linear scaling —  
     mllib\_MatrixScale\_S16\_S16\_Sat, 825  
 matrix linear scaling —  
     mllib\_MatrixScale\_S16\_S8\_Mod, 825  
 matrix linear scaling —  
     mllib\_MatrixScale\_S16\_S8\_Sat, 825  
 matrix linear scaling —  
     mllib\_MatrixScale\_S16\_U8\_Mod, 825  
 matrix linear scaling —  
     mllib\_MatrixScale\_S16\_U8\_Sat, 825  
 matrix linear scaling —  
     mllib\_MatrixScale\_S32C\_S16C\_Mod, 825  
 matrix linear scaling —  
     mllib\_MatrixScale\_S32C\_S16C\_Sat, 825  
 matrix linear scaling —  
     mllib\_MatrixScale\_S32C\_S32C\_Mod, 825  
 matrix linear scaling —  
     mllib\_MatrixScale\_S32C\_S32C\_Sat, 825  
 matrix linear scaling —  
     mllib\_MatrixScale\_S32\_S16\_Mod, 825  
 matrix linear scaling —  
     mllib\_MatrixScale\_S32\_S16\_Sat, 825  
 matrix linear scaling —  
     mllib\_MatrixScale\_S32\_S32\_Mod, 825  
 matrix linear scaling —  
     mllib\_MatrixScale\_S32\_S32\_Sat, 825  
 matrix linear scaling —  
     mllib\_MatrixScale\_S8C\_S8C\_Mod, 825  
 matrix linear scaling —  
     mllib\_MatrixScale\_S8C\_S8C\_Sat, 825  
 matrix linear scaling —  
     mllib\_MatrixScale\_S8\_S8\_Mod, 825  
 matrix linear scaling —  
     mllib\_MatrixScale\_S8\_S8\_Sat, 825  
 matrix linear scaling —  
     mllib\_MatrixScale\_U8C\_U8C\_Mod, 825  
 matrix linear scaling —  
     mllib\_MatrixScale\_U8C\_U8C\_Sat, 825  
 matrix linear scaling —  
     mllib\_MatrixScale\_U8\_U8\_Mod, 825  
 matrix linear scaling —  
     mllib\_MatrixScale\_U8\_U8\_Sat, 825  
 matrix linear scaling, in place —  
     mllib\_MatrixScale\_S16C\_Mod, 823  
 matrix linear scaling, in place —  
     mllib\_MatrixScale\_S16C\_Sat, 823  
 matrix linear scaling, in place —  
     mllib\_MatrixScale\_S16\_Mod, 823  
 matrix linear scaling, in place —  
     mllib\_MatrixScale\_S16\_Sat, 823  
 matrix linear scaling, in place —  
     mllib\_MatrixScale\_S32C\_Mod, 823  
 matrix linear scaling, in place —  
     mllib\_MatrixScale\_S32C\_Sat, 823  
 matrix linear scaling, in place —  
     mllib\_MatrixScale\_S32\_Mod, 823  
 matrix linear scaling, in place —  
     mllib\_MatrixScale\_S32\_Sat, 823  
 matrix linear scaling, in place —  
     mllib\_MatrixScale\_S8C\_Mod, 823  
 matrix linear scaling, in place —  
     mllib\_MatrixScale\_S8C\_Sat, 823  
 matrix linear scaling, in place —  
     mllib\_MatrixScale\_S8\_Mod, 823  
 matrix linear scaling, in place —  
     mllib\_MatrixScale\_S8\_Sat, 823  
 matrix linear scaling, in place —  
     mllib\_MatrixScale\_U8C\_Mod, 823  
 matrix linear scaling, in place —  
     mllib\_MatrixScale\_U8C\_Sat, 823  
 matrix linear scaling, in place —  
     mllib\_MatrixScale\_U8\_Mod, 823  
 matrix linear scaling, in place —  
     mllib\_MatrixScale\_U8\_Sat, 823  
 matrix multiplication —  
     mllib\_MatrixMul\_S16C\_S16C\_Mod, 819  
 matrix multiplication —  
     mllib\_MatrixMul\_S16C\_S16C\_Sat, 819  
 matrix multiplication —  
     mllib\_MatrixMul\_S16C\_S8C\_Mod, 819  
 matrix multiplication —  
     mllib\_MatrixMul\_S16C\_S8C\_Sat, 819  
 matrix multiplication —  
     mllib\_MatrixMul\_S16C\_U8C\_Mod, 819  
 matrix multiplication —  
     mllib\_MatrixMul\_S16C\_U8C\_Sat, 819  
 matrix multiplication —  
     mllib\_MatrixMul\_S16\_S16\_Mod, 819  
 matrix multiplication —  
     mllib\_MatrixMul\_S16\_S16\_Sat, 819  
 matrix multiplication —  
     mllib\_MatrixMul\_S16\_S8\_Mod, 819



matrix multiplication by scalar plus shifting —  
   mllib\_MatrixMulSShift\_S16C\_S16C\_Sat, 811  
 matrix multiplication by scalar plus shifting —  
   mllib\_MatrixMulSShift\_S16\_S16\_Mod, 811  
 matrix multiplication by scalar plus shifting —  
   mllib\_MatrixMulSShift\_S16\_S16\_Sat, 811  
 matrix multiplication by scalar plus shifting —  
   mllib\_MatrixMulSShift\_S32C\_S32C\_Mod, 811  
 matrix multiplication by scalar plus shifting —  
   mllib\_MatrixMulSShift\_S32C\_S32C\_Sat, 811  
 matrix multiplication by scalar plus shifting —  
   mllib\_MatrixMulSShift\_S32\_S32\_Mod, 811  
 matrix multiplication by scalar plus shifting —  
   mllib\_MatrixMulSShift\_S32\_S32\_Sat, 811  
 matrix multiplication by scalar plus shifting —  
   mllib\_MatrixMulSShift\_S8C\_S8C\_Mod, 811  
 matrix multiplication by scalar plus shifting —  
   mllib\_MatrixMulSShift\_S8C\_S8C\_Sat, 811  
 matrix multiplication by scalar plus shifting —  
   mllib\_MatrixMulSShift\_S8\_S8\_Mod, 811  
 matrix multiplication by scalar plus shifting —  
   mllib\_MatrixMulSShift\_S8\_S8\_Sat, 811  
 matrix multiplication by scalar plus shifting —  
   mllib\_MatrixMulSShift\_U8C\_U8C\_Mod, 811  
 matrix multiplication by scalar plus shifting —  
   mllib\_MatrixMulSShift\_U8C\_U8C\_Sat, 811  
 matrix multiplication by scalar plus shifting —  
   mllib\_MatrixMulSShift\_U8\_U8\_Mod, 811  
 matrix multiplication by scalar plus shifting —  
   mllib\_MatrixMulSShift\_U8\_U8\_Sat, 811  
 matrix multiplication by scalar plus shifting, in  
   place —  
   mllib\_MatrixMulSShift\_S16C\_Mod, 809  
 matrix multiplication by scalar plus shifting, in  
   place —  
   mllib\_MatrixMulSShift\_S16C\_Sat, 809  
 matrix multiplication by scalar plus shifting, in  
   place —  
   mllib\_MatrixMulSShift\_S16\_Mod, 809  
 matrix multiplication by scalar plus shifting, in  
   place — mllib\_MatrixMulSShift\_S16\_Sat, 809  
 matrix multiplication by scalar plus shifting, in  
   place —  
   mllib\_MatrixMulSShift\_S32C\_Mod, 809  
 matrix multiplication by scalar plus shifting, in  
   place —  
   mllib\_MatrixMulSShift\_S32C\_Sat, 809  
 matrix multiplication by scalar plus shifting, in  
   place —  
   mllib\_MatrixMulSShift\_S32\_Mod, 809  
 matrix multiplication by scalar plus shifting, in  
   place — mllib\_MatrixMulSShift\_S32\_Sat, 809  
 matrix multiplication by scalar plus shifting, in  
   place —  
   mllib\_MatrixMulSShift\_S8C\_Mod, 809  
 matrix multiplication by scalar plus shifting, in  
   place —  
   mllib\_MatrixMulSShift\_S8C\_Sat, 809  
 matrix multiplication by scalar plus shifting, in  
   place —  
   mllib\_MatrixMulSShift\_S8\_Mod, 809  
 matrix multiplication by scalar plus shifting, in  
   place — mllib\_MatrixMulSShift\_S8\_Sat, 809  
 matrix multiplication by scalar plus shifting, in  
   place —  
   mllib\_MatrixMulSShift\_U8C\_Mod, 809  
 matrix multiplication by scalar plus shifting, in  
   place —  
   mllib\_MatrixMulSShift\_U8C\_Sat, 809  
 matrix multiplication by scalar plus shifting, in  
   place —  
   mllib\_MatrixMulSShift\_U8\_Mod, 809  
 matrix multiplication by scalar plus shifting, in  
   place — mllib\_MatrixMulSShift\_U8\_Sat, 809  
 matrix multiplication by scalar, in place —  
   mllib\_MatrixMulS\_S16C\_Mod, 814  
 matrix multiplication by scalar, in place —  
   mllib\_MatrixMulS\_S16C\_Sat, 814  
 matrix multiplication by scalar, in place —  
   mllib\_MatrixMulS\_S16\_Mod, 814  
 matrix multiplication by scalar, in place —  
   mllib\_MatrixMulS\_S16\_Sat, 814  
 matrix multiplication by scalar, in place —  
   mllib\_MatrixMulS\_S32C\_Mod, 814  
 matrix multiplication by scalar, in place —  
   mllib\_MatrixMulS\_S32C\_Sat, 814  
 matrix multiplication by scalar, in place —  
   mllib\_MatrixMulS\_S32\_Mod, 814  
 matrix multiplication by scalar, in place —  
   mllib\_MatrixMulS\_S32\_Sat, 814  
 matrix multiplication by scalar, in place —  
   mllib\_MatrixMulS\_S8C\_Mod, 814  
 matrix multiplication by scalar, in place —  
   mllib\_MatrixMulS\_S8C\_Sat, 814

matrix multiplication by scalar, in place —  
     mllib\_MatrixMulS\_S8\_Mod, 814  
 matrix multiplication by scalar, in place —  
     mllib\_MatrixMulS\_S8\_Sat, 814  
 matrix multiplication by scalar, in place —  
     mllib\_MatrixMulS\_U8C\_Mod, 814  
 matrix multiplication by scalar, in place —  
     mllib\_MatrixMulS\_U8C\_Sat, 814  
 matrix multiplication by scalar, in place —  
     mllib\_MatrixMulS\_U8\_Mod, 814  
 matrix multiplication by scalar, in place —  
     mllib\_MatrixMulS\_U8\_Sat, 814  
 matrix multiplication plus shifting —  
     mllib\_MatrixMulShift\_S16C\_S16C\_Mod, 807  
 matrix multiplication plus shifting —  
     mllib\_MatrixMulShift\_S16C\_S16C\_Sat, 807  
 matrix multiplication plus shifting —  
     mllib\_MatrixMulShift\_S16\_S16\_Mod, 807  
 matrix multiplication plus shifting —  
     mllib\_MatrixMulShift\_S16\_S16\_Sat, 807  
 matrix subtraction —  
     mllib\_MatrixSub\_S16C\_S16C\_Mod, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_S16C\_S16C\_Sat, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_S16C\_S8C\_Mod, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_S16C\_S8C\_Sat, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_S16C\_U8C\_Mod, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_S16C\_U8C\_Sat, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_S16\_S16\_Mod, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_S16\_S16\_Sat, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_S16\_S8\_Mod, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_S16\_S8\_Sat, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_S16\_U8\_Mod, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_S16\_U8\_Sat, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_S32C\_S16C\_Mod, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_S32C\_S16C\_Sat, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_S32C\_S32C\_Mod, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_S32C\_S32C\_Sat, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_S32\_S16\_Mod, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_S32\_S16\_Sat, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_S32\_S32\_Mod, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_S32\_S32\_Sat, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_S8C\_S8C\_Mod, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_S8C\_S8C\_Sat, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_S8\_S8\_Mod, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_S8\_S8\_Sat, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_U8C\_U8C\_Mod, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_U8C\_U8C\_Sat, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_U8\_U8\_Mod, 836  
 matrix subtraction —  
     mllib\_MatrixSub\_U8\_U8\_Sat, 836  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_S16C\_S16C\_Mod, 831  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_S16C\_S16C\_Sat, 831  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_S16C\_S8C\_Mod, 831  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_S16C\_S8C\_Sat, 831  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_S16C\_U8C\_Mod, 831  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_S16C\_U8C\_Sat, 831  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_S16\_S16\_Mod, 831  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_S16\_S16\_Sat, 831  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_S16\_S8\_Mod, 831  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_S16\_S8\_Sat, 831

matrix subtraction from scalar —  
     mllib\_MatrixSubS\_S16\_U8\_Mod, 831  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_S16\_U8\_Sat, 831  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_S32C\_S16C\_Mod, 831  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_S32C\_S16C\_Sat, 831  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_S32C\_S32C\_Mod, 831  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_S32C\_S32C\_Sat, 831  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_S32\_S16\_Mod, 831  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_S32\_S16\_Sat, 831  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_S32\_S32\_Mod, 831  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_S32\_S32\_Sat, 831  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_S8C\_S8C\_Mod, 831  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_S8C\_S8C\_Sat, 831  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_S8\_S8\_Mod, 831  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_S8\_S8\_Sat, 831  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_U8C\_U8C\_Mod, 831  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_U8C\_U8C\_Sat, 831  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_U8\_U8\_Mod, 831  
 matrix subtraction from scalar —  
     mllib\_MatrixSubS\_U8\_U8\_Sat, 831  
 matrix subtraction from scalar, in place —  
     mllib\_MatrixSubS\_S16C\_Mod, 829  
 matrix subtraction from scalar, in place —  
     mllib\_MatrixSubS\_S16C\_Sat, 829  
 matrix subtraction from scalar, in place —  
     mllib\_MatrixSubS\_S16\_Mod, 829  
 matrix subtraction from scalar, in place —  
     mllib\_MatrixSubS\_S16\_Sat, 829  
 matrix subtraction from scalar, in place —  
     mllib\_MatrixSubS\_S32C\_Mod, 829  
 matrix subtraction from scalar, in place —  
     mllib\_MatrixSubS\_S32C\_Sat, 829  
 matrix subtraction from scalar, in place —  
     mllib\_MatrixSubS\_S32\_Mod, 829  
 matrix subtraction from scalar, in place —  
     mllib\_MatrixSubS\_S32\_Sat, 829  
 matrix subtraction from scalar, in place —  
     mllib\_MatrixSubS\_S8C\_Mod, 829  
 matrix subtraction from scalar, in place —  
     mllib\_MatrixSubS\_S8C\_Sat, 829  
 matrix subtraction from scalar, in place —  
     mllib\_MatrixSubS\_S8\_Mod, 829  
 matrix subtraction from scalar, in place —  
     mllib\_MatrixSubS\_S8\_Sat, 829  
 matrix subtraction from scalar, in place —  
     mllib\_MatrixSubS\_U8C\_Mod, 829  
 matrix subtraction from scalar, in place —  
     mllib\_MatrixSubS\_U8C\_Sat, 829  
 matrix subtraction from scalar, in place —  
     mllib\_MatrixSubS\_U8\_Mod, 829  
 matrix subtraction from scalar, in place —  
     mllib\_MatrixSubS\_U8\_Sat, 829  
 matrix subtraction, in place —  
     mllib\_MatrixSub\_S16C\_Mod, 834  
 matrix subtraction, in place —  
     mllib\_MatrixSub\_S16C\_Sat, 834  
 matrix subtraction, in place —  
     mllib\_MatrixSub\_S16\_Mod, 834  
 matrix subtraction, in place —  
     mllib\_MatrixSub\_S16\_Sat, 834  
 matrix subtraction, in place —  
     mllib\_MatrixSub\_S32C\_Mod, 834  
 matrix subtraction, in place —  
     mllib\_MatrixSub\_S32C\_Sat, 834  
 matrix subtraction, in place —  
     mllib\_MatrixSub\_S32\_Mod, 834  
 matrix subtraction, in place —  
     mllib\_MatrixSub\_S32\_Sat, 834  
 matrix subtraction, in place —  
     mllib\_MatrixSub\_S8C\_Mod, 834  
 matrix subtraction, in place —  
     mllib\_MatrixSub\_S8C\_Sat, 834  
 matrix subtraction, in place —  
     mllib\_MatrixSub\_S8\_Mod, 834  
 matrix subtraction, in place —  
     mllib\_MatrixSub\_S8\_Sat, 834  
 matrix subtraction, in place —  
     mllib\_MatrixSub\_U8C\_Mod, 834  
 matrix subtraction, in place —  
     mllib\_MatrixSub\_U8C\_Sat, 834



matrix subtraction, in place —  
     mllib\_MatrixSub\_U8\_Mod, 834  
 matrix subtraction, in place —  
     mllib\_MatrixSub\_U8\_Sat, 834  
 matrix transpose —  
     mllib\_MatrixTranspose\_S16C\_S16C, 841  
 matrix transpose —  
     mllib\_MatrixTranspose\_S16\_S16, 841  
 matrix transpose —  
     mllib\_MatrixTranspose\_S32C\_S32C, 841  
 matrix transpose —  
     mllib\_MatrixTranspose\_S32\_S32, 841  
 matrix transpose —  
     mllib\_MatrixTranspose\_S8C\_S8C, 841  
 matrix transpose —  
     mllib\_MatrixTranspose\_S8\_S8, 841  
 matrix transpose —  
     mllib\_MatrixTranspose\_U8C\_U8C, 841  
 matrix transpose —  
     mllib\_MatrixTranspose\_U8\_U8, 841  
 matrix transpose, in place —  
     mllib\_MatrixTranspose\_S16, 839  
 matrix transpose, in place —  
     mllib\_MatrixTranspose\_S16C, 839  
 matrix transpose, in place —  
     mllib\_MatrixTranspose\_S32, 839  
 matrix transpose, in place —  
     mllib\_MatrixTranspose\_S32C, 839  
 matrix transpose, in place —  
     mllib\_MatrixTranspose\_S8, 839  
 matrix transpose, in place —  
     mllib\_MatrixTranspose\_S8C, 839  
 matrix transpose, in place —  
     mllib\_MatrixTranspose\_U8, 839  
 matrix transpose, in place —  
     mllib\_MatrixTranspose\_U8C, 839  
 maximum intensity searching —  
     mllib\_VolumeFindMaxBMask\_S16, 1538  
 maximum intensity searching —  
     mllib\_VolumeFindMaxBMask\_U8, 1538  
 maximum intensity searching —  
     mllib\_VolumeFindMaxCMask\_S16, 1539  
 maximum intensity searching —  
     mllib\_VolumeFindMaxCMask\_U8, 1539  
 maximum intensity searching —  
     mllib\_VolumeFindMax\_S16, 1540  
 maximum intensity searching —  
     mllib\_VolumeFindMax\_U8, 1540  
 merge — mllib\_SignalMerge\_F32S\_F32, 1108  
 merge — mllib\_SignalMerge\_S16S\_S16, 1109  
 mllib\_free — free a block of bytes, 40  
 mllib\_GraphicsBoundaryFill\_32 — boundary  
     fill, 41  
 mllib\_GraphicsBoundaryFill\_8 — boundary  
     fill, 41  
 mllib\_GraphicsDrawArc\_32 — draw arc, 42  
 mllib\_GraphicsDrawArc\_8 — draw arc, 42  
 mllib\_GraphicsDrawArc\_A\_32 — draw arc with  
     antialiasing, 43  
 mllib\_GraphicsDrawArc\_A\_8 — draw arc with  
     antialiasing, 43  
 mllib\_GraphicsDrawArc\_X\_32 — draw arc in  
     Xor mode, 44  
 mllib\_GraphicsDrawArc\_X\_8 — draw arc in Xor  
     mode, 44  
 mllib\_GraphicsDrawCircle\_32 — draw circle, 45  
 mllib\_GraphicsDrawCircle\_8 — draw circle, 45  
 mllib\_GraphicsDrawCircle\_A\_32 — draw circle  
     with antialiasing, 46  
 mllib\_GraphicsDrawCircle\_A\_8 — draw circle  
     with antialiasing, 46  
 mllib\_GraphicsDrawCircle\_X\_32 — draw circle  
     in Xor mode, 47  
 mllib\_GraphicsDrawCircle\_X\_8 — draw circle in  
     Xor mode, 47  
 mllib\_GraphicsDrawEllipse\_32 — draw  
     ellipse, 48  
 mllib\_GraphicsDrawEllipse\_8 — draw  
     ellipse, 48  
 mllib\_GraphicsDrawEllipse\_A\_32 — draw  
     ellipse with antialiasing, 49  
 mllib\_GraphicsDrawEllipse\_A\_8 — draw ellipse  
     with antialiasing, 49  
 mllib\_GraphicsDrawEllipse\_X\_32 — draw  
     ellipse in Xor mode, 50  
 mllib\_GraphicsDrawEllipse\_X\_8 — draw ellipse  
     in Xor mode, 50  
 mllib\_GraphicsDrawLine\_32 — draw line, 51  
 mllib\_GraphicsDrawLine\_8 — draw line, 51  
 mllib\_GraphicsDrawLine\_A\_32 — draw line  
     with antialiasing, 52  
 mllib\_GraphicsDrawLine\_A\_8 — draw line with  
     antialiasing, 52  
 mllib\_GraphicsDrawLine\_AG\_32 — draw line  
     with antialiasing and Gouraud shading, 53

`mllib_GraphicsDrawLine_AG_8` — draw line with antialiasing and Gouraud shading, 53  
`mllib_GraphicsDrawLine_AGZ_32` — draw line with antialiasing, Gouraud shading, and Z buffering, 54  
`mllib_GraphicsDrawLine_AGZ_8` — draw line with antialiasing, Gouraud shading, and Z buffering, 54  
`mllib_GraphicsDrawLine_AZ_32` — draw line with antialiasing and Z buffering, 56  
`mllib_GraphicsDrawLine_AZ_8` — draw line with antialiasing and Z buffering, 56  
`mllib_GraphicsDrawLine_G_32` — draw line with Gouraud shading, 66  
`mllib_GraphicsDrawLine_G_8` — draw line with Gouraud shading, 66  
`mllib_GraphicsDrawLine_GZ_32` — draw line with Gouraud shading and Z buffering, 67  
`mllib_GraphicsDrawLine_GZ_8` — draw line with Gouraud shading and Z buffering, 67  
`mllib_GraphicsDrawLine_X_32` — draw line in Xor mode, 87  
`mllib_GraphicsDrawLine_X_8` — draw line in Xor mode, 87  
`mllib_GraphicsDrawLine_Z_32` — draw line with Z buffering, 88  
`mllib_GraphicsDrawLine_Z_8` — draw line with Z buffering, 88  
`mllib_GraphicsDrawLineFanSet_32` — draw line set where all members of the set have one common end point, 57  
`mllib_GraphicsDrawLineFanSet_8` — draw line set where all members of the set have one common end point, 57  
`mllib_GraphicsDrawLineFanSet_A_32` — draw line set with antialiasing, where all members of the set have one common end point, 58  
`mllib_GraphicsDrawLineFanSet_A_8` — draw line set with antialiasing, where all members of the set have one common end point, 58  
`mllib_GraphicsDrawLineFanSet_AG_32` — draw line set with antialiasing and gouraud shading, where all members of the set have one common end point, 59  
`mllib_GraphicsDrawLineFanSet_AG_8` — draw line set with antialiasing and gouraud shading, where all members of the set have one common end point, 59  
`mllib_GraphicsDrawLineFanSet_AGZ_32` — draw line set with antialiasing, Gouraud shading, and Z buffering, where all members of the set have one common end point, 60  
`mllib_GraphicsDrawLineFanSet_AGZ_8` — draw line set with antialiasing, Gouraud shading, and Z buffering, where all members of the set have one common end point, 60  
`mllib_GraphicsDrawLineFanSet_AZ_32` — draw line set with antialiasing and Z buffering, where all members of the set have one common end point, 61  
`mllib_GraphicsDrawLineFanSet_AZ_8` — draw line set with antialiasing and Z buffering, where all members of the set have one common end point, 61  
`mllib_GraphicsDrawLineFanSet_G_32` — draw line set with Gouraud shading, where all members of the set have one common end point, 62  
`mllib_GraphicsDrawLineFanSet_G_8` — draw line set with Gouraud shading, where all members of the set have one common end point, 62  
`mllib_GraphicsDrawLineFanSet_GZ_32` — draw line set with Gouraud shading and Z buffering, where all members of the set have one common end point, 63  
`mllib_GraphicsDrawLineFanSet_GZ_8` — draw line set with Gouraud shading and Z buffering, where all members of the set have one common end point, 63  
`mllib_GraphicsDrawLineFanSet_X_32` — draw line set in Xor mode where all members of the set have one common end point, 64  
`mllib_GraphicsDrawLineFanSet_X_8` — draw line set in Xor mode where all members of the set have one common end point, 64  
`mllib_GraphicsDrawLineFanSet_Z_32` — draw line set with Z buffering where all members of the set have one common end point, 65  
`mllib_GraphicsDrawLineFanSet_Z_8` — draw line set with Z buffering where all members of the set have one common end point, 65  
`mllib_GraphicsDrawLineSet_32` — draw line set where each member can have different end points, 69

`mllib_GraphicsDrawLineSet_8` — draw line set where each member can have different end points, 69

`mllib_GraphicsDrawLineSet_A_32` — draw line set with antialiasing where each member can have different end points, 70

`mllib_GraphicsDrawLineSet_A_8` — draw line set with antialiasing where each member can have different end points, 70

`mllib_GraphicsDrawLineSet_AG_32` — draw line set with antialiasing and Gouraud shading where each member can have different end points, 71

`mllib_GraphicsDrawLineSet_AG_8` — draw line set with antialiasing and Gouraud shading where each member can have different end points, 71

`mllib_GraphicsDrawLineSet_AGZ_32` — draw line set with antialiasing, Gouraud shading, and Z buffering, where each member can have different end points, 72

`mllib_GraphicsDrawLineSet_AGZ_8` — draw line set with antialiasing, Gouraud shading, and Z buffering, where each member can have different end points, 72

`mllib_GraphicsDrawLineSet_AZ_32` — draw line set with antialiasing and Z buffering, where each member can have different end points, 73

`mllib_GraphicsDrawLineSet_AZ_8` — draw line set with antialiasing and Z buffering, where each member can have different end points, 73

`mllib_GraphicsDrawLineSet_G_32` — draw line set with Gouraud shading where each member can have different end points, 74

`mllib_GraphicsDrawLineSet_G_8` — draw line set with Gouraud shading where each member can have different end points, 74

`mllib_GraphicsDrawLineSet_GZ_32` — draw line set with Gouraud shading and Z buffering, where each member can have different end points, 75

`mllib_GraphicsDrawLineSet_GZ_8` — draw line set with Gouraud shading and Z buffering, where each member can have different end points, 75

`mllib_GraphicsDrawLineSet_X_32` — draw line set in Xor mode where each member can have different end points, 76

`mllib_GraphicsDrawLineSet_X_8` — draw line set in Xor mode where each member can have different end points, 76

`mllib_GraphicsDrawLineSet_Z_32` — draw line set with Z buffering where each member can have different end points, 77

`mllib_GraphicsDrawLineSet_Z_8` — draw line set with Z buffering where each member can have different end points, 77

`mllib_GraphicsDrawLineStripSet_32` — draw line set where each member of the set starts at the point where the previous member ended, 78

`mllib_GraphicsDrawLineStripSet_8` — draw line set where each member of the set starts at the point where the previous member ended, 78

`mllib_GraphicsDrawLineStripSet_A_32` — draw line set with antialiasing where each member of the set starts at the point where the previous member ended, 79

`mllib_GraphicsDrawLineStripSet_A_8` — draw line set with antialiasing where each member of the set starts at the point where the previous member ended, 79

`mllib_GraphicsDrawLineStripSet_AG_32` — draw line set with antialiasing and gouraud shading where each member of the set starts at the point where the previous member ended, 80

`mllib_GraphicsDrawLineStripSet_AG_8` — draw line set with antialiasing and gouraud shading where each member of the set starts at the point where the previous member ended, 80

`mllib_GraphicsDrawLineStripSet_AGZ_32` — draw line set with antialiasing, Gouraud shading, and Z buffering, where each member of the set starts at the point where the previous member ended, 81

`mllib_GraphicsDrawLineStripSet_AGZ_8` — draw line set with antialiasing, Gouraud shading, and Z buffering, where each member of the set starts at the point where the previous member ended, 81

mlib\_GraphicsDrawLineStripSet\_AZ\_32 — draw line set with antialiasing and Z buffering, where each member of the set starts at the point where the previous member ended, 82  
 mlib\_GraphicsDrawLineStripSet\_AZ\_8 — draw line set with antialiasing and Z buffering, where each member of the set starts at the point where the previous member ended, 82  
 mlib\_GraphicsDrawLineStripSet\_G\_32 — draw line set with Gouraud shading, where each member of the set starts at the point where the previous member ended, 83  
 mlib\_GraphicsDrawLineStripSet\_G\_8 — draw line set with Gouraud shading, where each member of the set starts at the point where the previous member ended, 83  
 mlib\_GraphicsDrawLineStripSet\_GZ\_32 — draw line set with Gouraud shading and Z buffering, where each member of the set starts at the point where the previous member ended, 84  
 mlib\_GraphicsDrawLineStripSet\_GZ\_8 — draw line set with Gouraud shading and Z buffering, where each member of the set starts at the point where the previous member ended, 84  
 mlib\_GraphicsDrawLineStripSet\_X\_32 — draw line set in Xor mode where each member of the set starts at the point where the previous member ended, 85  
 mlib\_GraphicsDrawLineStripSet\_X\_8 — draw line set in Xor mode where each member of the set starts at the point where the previous member ended, 85  
 mlib\_GraphicsDrawLineStripSet\_Z\_32 — draw line set with Z buffering where each member of the set starts at the point where the previous member ended, 86  
 mlib\_GraphicsDrawLineStripSet\_Z\_8 — draw line set with Z buffering where each member of the set starts at the point where the previous member ended, 86  
 mlib\_GraphicsDrawPoint\_32 — draw point, 89  
 mlib\_GraphicsDrawPoint\_8 — draw point, 89  
 mlib\_GraphicsDrawPoint\_X\_32 — draw point in Xor mode, 92  
 mlib\_GraphicsDrawPoint\_X\_8 — draw point in Xor mode, 92  
 mlib\_GraphicsDrawPointSet\_32 — draw point set where each member can be a different point, 90  
 mlib\_GraphicsDrawPointSet\_8 — draw point set where each member can be a different point, 90  
 mlib\_GraphicsDrawPointSet\_X\_32 — draw point set in Xor mode where each member can be a different point, 91  
 mlib\_GraphicsDrawPointSet\_X\_8 — draw point set in Xor mode where each member can be a different point, 91  
 mlib\_GraphicsDrawPolygon\_32 — draw polygon, 93  
 mlib\_GraphicsDrawPolygon\_8 — draw polygon, 93  
 mlib\_GraphicsDrawPolygon\_A\_32 — draw polygon with antialiasing, 94  
 mlib\_GraphicsDrawPolygon\_A\_8 — draw polygon with antialiasing, 94  
 mlib\_GraphicsDrawPolygon\_AG\_32 — draw line with antialiasing and Gouraud shading, 95  
 mlib\_GraphicsDrawPolygon\_AG\_8 — draw line with antialiasing and Gouraud shading, 95  
 mlib\_GraphicsDrawPolygon\_AGZ\_32 — draw polygon with antialiasing, Gouraud shading, and Z buffering, 96  
 mlib\_GraphicsDrawPolygon\_AGZ\_8 — draw polygon with antialiasing, Gouraud shading, and Z buffering, 96  
 mlib\_GraphicsDrawPolygon\_AZ\_32 — draw polygon with antialiasing and Z buffering, 97  
 mlib\_GraphicsDrawPolygon\_AZ\_8 — draw polygon with antialiasing and Z buffering, 97  
 mlib\_GraphicsDrawPolygon\_G\_32 — draw polygon with Gouraud shading, 98  
 mlib\_GraphicsDrawPolygon\_G\_8 — draw polygon with Gouraud shading, 98  
 mlib\_GraphicsDrawPolygon\_GZ\_32 — draw polygon with Gouraud shading and Z buffering, 99

mlib\_GraphicsDrawPolygon\_GZ\_8 — draw polygon with Gouraud shading and Z buffering, 99  
 mlib\_GraphicsDrawPolygon\_X\_32 — draw polygon in Xor mode, 100  
 mlib\_GraphicsDrawPolygon\_X\_8 — draw polygon in Xor mode, 100  
 mlib\_GraphicsDrawPolygon\_Z\_32 — draw polygon with Z buffering, 101  
 mlib\_GraphicsDrawPolygon\_Z\_8 — draw polygon with Z buffering, 101  
 mlib\_GraphicsDrawPolyline\_32 — draw polyline, 102  
 mlib\_GraphicsDrawPolyline\_8 — draw polyline, 102  
 mlib\_GraphicsDrawPolyline\_A\_32 — draw polyline with antialiasing, 103  
 mlib\_GraphicsDrawPolyline\_A\_8 — draw polyline with antialiasing, 103  
 mlib\_GraphicsDrawPolyline\_AG\_32 — draw line with antialiasing and Gouraud shading, 104  
 mlib\_GraphicsDrawPolyline\_AG\_8 — draw line with antialiasing and Gouraud shading, 104  
 mlib\_GraphicsDrawPolyline\_AGZ\_32 — draw polyline with antialiasing, Gouraud shading, and Z buffering, 105  
 mlib\_GraphicsDrawPolyline\_AGZ\_8 — draw polyline with antialiasing, Gouraud shading, and Z buffering, 105  
 mlib\_GraphicsDrawPolyline\_AZ\_32 — draw polyline with antialiasing and Z buffering, 106  
 mlib\_GraphicsDrawPolyline\_AZ\_8 — draw polyline with antialiasing and Z buffering, 106  
 mlib\_GraphicsDrawPolyline\_G\_32 — draw polyline with Gouraud shading, 107  
 mlib\_GraphicsDrawPolyline\_G\_8 — draw polyline with Gouraud shading, 107  
 mlib\_GraphicsDrawPolyline\_GZ\_32 — draw polyline with Gouraud shading and Z buffering, 108  
 mlib\_GraphicsDrawPolyline\_GZ\_8 — draw polyline with Gouraud shading and Z buffering, 108  
 mlib\_GraphicsDrawPolyline\_X\_32 — draw polyline in Xor mode, 109  
 mlib\_GraphicsDrawPolyline\_X\_8 — draw polyline in Xor mode, 109  
 mlib\_GraphicsDrawPolyline\_Z\_32 — draw polyline with Z buffering, 110  
 mlib\_GraphicsDrawPolyline\_Z\_8 — draw polyline with Z buffering, 110  
 mlib\_GraphicsDrawPolypoint\_32 — draw polypoint, 111  
 mlib\_GraphicsDrawPolypoint\_8 — draw polypoint, 111  
 mlib\_GraphicsDrawPolypoint\_X\_32 — draw polypoint in Xor mode, 112  
 mlib\_GraphicsDrawPolypoint\_X\_8 — draw polypoint in Xor mode, 112  
 mlib\_GraphicsDrawRectangle\_32 — draw rectangle, 113  
 mlib\_GraphicsDrawRectangle\_8 — draw rectangle, 113  
 mlib\_GraphicsDrawRectangle\_X\_32 — draw rectangle in Xor mode, 114  
 mlib\_GraphicsDrawRectangle\_X\_8 — draw rectangle in Xor mode, 114  
 mlib\_GraphicsDrawTriangle\_32 — draw triangle, 115  
 mlib\_GraphicsDrawTriangle\_8 — draw triangle, 115  
 mlib\_GraphicsDrawTriangle\_A\_32 — draw triangle with antialiasing, 116  
 mlib\_GraphicsDrawTriangle\_A\_8 — draw triangle with antialiasing, 116  
 mlib\_GraphicsDrawTriangle\_AG\_32 — draw triangle with antialiasing and Gouraud shading, 117  
 mlib\_GraphicsDrawTriangle\_AG\_8 — draw triangle with antialiasing and Gouraud shading, 117  
 mlib\_GraphicsDrawTriangle\_AGZ\_32 — draw triangle with antialiasing, Gouraud shading, and Z buffering, 119  
 mlib\_GraphicsDrawTriangle\_AGZ\_8 — draw triangle with antialiasing, Gouraud shading, and Z buffering, 119  
 mlib\_GraphicsDrawTriangle\_AZ\_32 — draw triangle with antialiasing and Z buffering, 121

`mllib_GraphicsDrawTriangle_AZ_8` — draw triangle with antialiasing and Z buffering, 121

`mllib_GraphicsDrawTriangle_G_32` — draw triangle with Gouraud shading, 132

`mllib_GraphicsDrawTriangle_G_8` — draw triangle with Gouraud shading, 132

`mllib_GraphicsDrawTriangle_GZ_32` — draw triangle with Gouraud shading and Z buffering, 134

`mllib_GraphicsDrawTriangle_GZ_8` — draw triangle with Gouraud shading and Z buffering, 134

`mllib_GraphicsDrawTriangle_X_32` — draw triangle in Xor mode, 155

`mllib_GraphicsDrawTriangle_X_8` — draw triangle in Xor mode, 155

`mllib_GraphicsDrawTriangle_Z_32` — draw triangle with Z buffering, 157

`mllib_GraphicsDrawTriangle_Z_8` — draw triangle with Z buffering, 157

`mllib_GraphicsDrawTriangleFanSet_32` — draw triangle set where all members of the set have a common vertex, 123

`mllib_GraphicsDrawTriangleFanSet_8` — draw triangle set where all members of the set have a common vertex, 123

`mllib_GraphicsDrawTriangleFanSet_A_32` — draw triangle set with antialiasing where all members of the set have a common vertex, 124

`mllib_GraphicsDrawTriangleFanSet_A_8` — draw triangle set with antialiasing where all members of the set have a common vertex, 124

`mllib_GraphicsDrawTriangleFanSet_AG_32` — draw triangle set with antialiasing and gouraud shading, where all members of the set have a common vertex, 125

`mllib_GraphicsDrawTriangleFanSet_AG_8` — draw triangle set with antialiasing and gouraud shading, where all members of the set have a common vertex, 125

`mllib_GraphicsDrawTriangleFanSet_AGZ_32` — draw triangle set with antialiasing, Gouraud shading, and Z buffering, where all members of the set have a common vertex, 126

`mllib_GraphicsDrawTriangleFanSet_AGZ_8` — draw triangle set with antialiasing, Gouraud shading, and Z buffering, where all members of the set have a common vertex, 126

`mllib_GraphicsDrawTriangleFanSet_AZ_32` — draw triangle set with antialiasing and Z buffering, where all members of the set have a common vertex, 127

`mllib_GraphicsDrawTriangleFanSet_AZ_8` — draw triangle set with antialiasing and Z buffering, where all members of the set have a common vertex, 127

`mllib_GraphicsDrawTriangleFanSet_G_32` — draw triangle set with Gouraud shading where all members of the set have a common vertex, 128

`mllib_GraphicsDrawTriangleFanSet_G_8` — draw triangle set with Gouraud shading where all members of the set have a common vertex, 128

`mllib_GraphicsDrawTriangleFanSet_GZ_32` — draw triangle set with Gouraud shading and Z buffering, where all members of the set have a common vertex, 129

`mllib_GraphicsDrawTriangleFanSet_GZ_8` — draw triangle set with Gouraud shading and Z buffering, where all members of the set have a common vertex, 129

`mllib_GraphicsDrawTriangleFanSet_X_32` — draw triangle set in Xor mode where all members of the set have a common vertex, 130

`mllib_GraphicsDrawTriangleFanSet_X_8` — draw triangle set in Xor mode where all members of the set have a common vertex, 130

`mllib_GraphicsDrawTriangleFanSet_Z_32` — draw triangle set with Z buffering where all members of the set have a common vertex, 131

`mllib_GraphicsDrawTriangleFanSet_Z_8` — draw triangle set with Z buffering where all members of the set have a common vertex, 131

`mllib_GraphicsDrawTriangleSet_32` — draw triangle set where each member can have different vertices, 136

`mllib_GraphicsDrawTriangleSet_8` — draw triangle set where each member can have different vertices, 136

`mllib_GraphicsDrawTriangleSet_A_32` — draw triangle set with antialiasing where each member can have different vertices, 137

`mllib_GraphicsDrawTriangleSet_A_8` — draw triangle set with antialiasing where each member can have different vertices, 137

`mllib_GraphicsDrawTriangleSet_AG_32` — draw triangle set with antialiasing and Gouraud shading, where each member can have different vertices, 138

`mllib_GraphicsDrawTriangleSet_AG_8` — draw triangle set with antialiasing and Gouraud shading, where each member can have different vertices, 138

`mllib_GraphicsDrawTriangleSet_AGZ_32` — draw triangle set with antialiasing, Gouraud shading, and Z buffering, where each member can have different vertices, 139

`mllib_GraphicsDrawTriangleSet_AGZ_8` — draw triangle set with antialiasing, Gouraud shading, and Z buffering, where each member can have different vertices, 139

`mllib_GraphicsDrawTriangleSet_AZ_32` — draw triangle set with antialiasing and Z buffering, where each member can have different vertices, 140

`mllib_GraphicsDrawTriangleSet_AZ_8` — draw triangle set with antialiasing and Z buffering, where each member can have different vertices, 140

`mllib_GraphicsDrawTriangleSet_G_32` — draw triangle set with Gouraud shading where each member can have different vertices, 141

`mllib_GraphicsDrawTriangleSet_G_8` — draw triangle set with Gouraud shading where each member can have different vertices, 141

`mllib_GraphicsDrawTriangleSet_GZ_32` — draw triangle set with Gouraud shading and Z buffering, where each member can have different vertices, 142

`mllib_GraphicsDrawTriangleSet_GZ_8` — draw triangle set with Gouraud shading and Z buffering, where each member can have different vertices, 142

`mllib_GraphicsDrawTriangleSet_X_32` — draw triangle set in Xor mode where each member can have different vertices, 143

`mllib_GraphicsDrawTriangleSet_X_8` — draw triangle set in Xor mode where each member can have different vertices, 143

`mllib_GraphicsDrawTriangleSet_Z_32` — draw triangle set with Z buffering, 144

`mllib_GraphicsDrawTriangleSet_Z_8` — draw triangle set with Z buffering, 144

`mllib_GraphicsDrawTriangleStripSet_32` — draw triangle set where the first side of each member is common to the second side of the previous member, 145

`mllib_GraphicsDrawTriangleStripSet_8` — draw triangle set where the first side of each member is common to the second side of the previous member, 145

`mllib_GraphicsDrawTriangleStripSet_A_32` — draw triangle set with antialiasing where the first side of each member is common to the second side of the previous member, 146

`mllib_GraphicsDrawTriangleStripSet_A_8` — draw triangle set with antialiasing where the first side of each member is common to the second side of the previous member, 146

`mllib_GraphicsDrawTriangleStripSet_AG_32` — draw triangle set with antialiasing and gouraud shading, 147

`mllib_GraphicsDrawTriangleStripSet_AG_8` — draw triangle set with antialiasing and gouraud shading, 147

`mllib_GraphicsDrawTriangleStripSet_AGZ` — draw triangle set with antialiasing, Gouraud shading, and Z buffering, where the first side of each member is common to the second side of the previous member, 148

`mllib_GraphicsDrawTriangleStripSet_AGZ_32` — draw triangle set with antialiasing, Gouraud shading, and Z buffering, where the first side of each member is common to the second side of the previous member, 148

`mllib_GraphicsDrawTriangleStripSet_AGZ_8` — draw triangle set with antialiasing, Gouraud shading, and Z buffering, where the first side of each member is common to the second side of the previous member, 148

`mllib_GraphicsDrawTriangleStripSet_AZ_32` — draw triangle set with antialiasing and Z buffering, where the first side of each member is common to the second side of the previous member, 150  
`mllib_GraphicsDrawTriangleStripSet_AZ_8` — draw triangle set with antialiasing and Z buffering, where the first side of each member is common to the second side of the previous member, 150  
`mllib_GraphicsDrawTriangleStripSet_G_32` — draw triangle set with Gouraud shading where the first side of each member is common to the second side of the previous member, 151  
`mllib_GraphicsDrawTriangleStripSet_G_8` — draw triangle set with Gouraud shading where the first side of each member is common to the second side of the previous member, 151  
`mllib_GraphicsDrawTriangleStripSet_GZ_32` — draw triangle set with Gouraud shading and Z buffering, where the first side of each member is common to the second side of the previous member, 152  
`mllib_GraphicsDrawTriangleStripSet_GZ_8` — draw triangle set with Gouraud shading and Z buffering, where the first side of each member is common to the second side of the previous member, 152  
`mllib_GraphicsDrawTriangleStripSet_X_32` — draw triangle set in Xor mode where the first side of each member is common to the second side of the previous member, 153  
`mllib_GraphicsDrawTriangleStripSet_X_8` — draw triangle set in Xor mode where the first side of each member is common to the second side of the previous member, 153  
`mllib_GraphicsDrawTriangleStripSet_Z_32` — draw triangle set with Z buffering where the first side of each member is common to the second side of the previous member, 154  
`mllib_GraphicsDrawTriangleStripSet_Z_8` — draw triangle set with Z buffering where the first side of each member is common to the second side of the previous member, 154  
`mllib_GraphicsFillArc_32` — draw filled arc, 159  
`mllib_GraphicsFillArc_8` — draw filled arc, 159  
`mllib_GraphicsFillArc_A_32` — draw filled arc with antialiasing, 160  
`mllib_GraphicsFillArc_A_8` — draw filled arc with antialiasing, 160  
`mllib_GraphicsFillArc_X_32` — draw filled arc in Xor mode, 161  
`mllib_GraphicsFillArc_X_8` — draw filled arc in Xor mode, 161  
`mllib_GraphicsFillCircle_32` — draw filled circle, 162  
`mllib_GraphicsFillCircle_8` — draw filled circle, 162  
`mllib_GraphicsFillCircle_A_32` — draw filled circle with antialiasing, 163  
`mllib_GraphicsFillCircle_A_8` — draw filled circle with antialiasing, 163  
`mllib_GraphicsFillCircle_X_32` — draw filled circle in Xor mode, 164  
`mllib_GraphicsFillCircle_X_8` — draw filled circle in Xor mode, 164  
`mllib_GraphicsFillEllipse_32` — draw filled ellipse, 165  
`mllib_GraphicsFillEllipse_8` — draw filled ellipse, 165  
`mllib_GraphicsFillEllipse_A_32` — draw filled ellipse with antialiasing, 166  
`mllib_GraphicsFillEllipse_A_8` — draw filled ellipse with antialiasing, 166  
`mllib_GraphicsFillEllipse_X_32` — draw filled ellipse in Xor mode, 167  
`mllib_GraphicsFillEllipse_X_8` — draw filled ellipse in Xor mode, 167  
`mllib_GraphicsFillPolygon_32` — draw filled polygon, 168  
`mllib_GraphicsFillPolygon_8` — draw filled polygon, 168  
`mllib_GraphicsFillPolygon_A_32` — draw filled polygon with antialiasing, 169  
`mllib_GraphicsFillPolygon_A_8` — draw filled polygon with antialiasing, 169  
`mllib_GraphicsFillPolygon_AG_32` — draw filled polygon with antialiasing and Gouraud shading, 170  
`mllib_GraphicsFillPolygon_AG_8` — draw filled polygon with antialiasing and Gouraud shading, 170



mlib\_GraphicsFillPolygon\_AGZ\_32 — draw filled polygon with antialiasing, Gouraud shading, and Z buffering, 171  
 mlib\_GraphicsFillPolygon\_AGZ\_8 — draw filled polygon with antialiasing, Gouraud shading, and Z buffering, 171  
 mlib\_GraphicsFillPolygon\_AZ\_32 — draw filled polygon with antialiasing and Z buffering, 172  
 mlib\_GraphicsFillPolygon\_AZ\_8 — draw filled polygon with antialiasing and Z buffering, 172  
 mlib\_GraphicsFillPolygon\_G\_32 — draw filled polygon with Gouraud shading, 173  
 mlib\_GraphicsFillPolygon\_G\_8 — draw filled polygon with Gouraud shading, 173  
 mlib\_GraphicsFillPolygon\_GZ\_32 — draw filled polygon with Gouraud shading and Z buffering, 174  
 mlib\_GraphicsFillPolygon\_GZ\_8 — draw filled polygon with Gouraud shading and Z buffering, 174  
 mlib\_GraphicsFillPolygon\_X\_32 — draw filled polygon in Xor mode, 175  
 mlib\_GraphicsFillPolygon\_X\_8 — draw filled polygon in Xor mode, 175  
 mlib\_GraphicsFillPolygon\_Z\_32 — draw filled polygon with Z buffering, 176  
 mlib\_GraphicsFillPolygon\_Z\_8 — draw filled polygon with Z buffering, 176  
 mlib\_GraphicsFillRectangle\_32 — draw filled rectangle, 177  
 mlib\_GraphicsFillRectangle\_8 — draw filled rectangle, 177  
 mlib\_GraphicsFillRectangle\_X\_32 — draw filled rectangle in Xor mode, 178  
 mlib\_GraphicsFillRectangle\_X\_8 — draw filled rectangle in Xor mode, 178  
 mlib\_GraphicsFillTriangle\_32 — draw filled triangle, 179  
 mlib\_GraphicsFillTriangle\_8 — draw filled triangle, 179  
 mlib\_GraphicsFillTriangle\_A\_32 — draw filled triangle with antialiasing, 180  
 mlib\_GraphicsFillTriangle\_A\_8 — draw filled triangle with antialiasing, 180  
 mlib\_GraphicsFillTriangle\_AG\_32 — draw line with antialiasing and Gouraud shading, 181  
 mlib\_GraphicsFillTriangle\_AG\_8 — draw line with antialiasing and Gouraud shading, 181  
 mlib\_GraphicsFillTriangle\_AGZ\_32 — draw filled triangle with antialiasing, Gouraud shading, and Z buffering, 183  
 mlib\_GraphicsFillTriangle\_AGZ\_8 — draw filled triangle with antialiasing, Gouraud shading, and Z buffering, 183  
 mlib\_GraphicsFillTriangle\_AZ\_32 — draw filled triangle with antialiasing and Z buffering, 185  
 mlib\_GraphicsFillTriangle\_AZ\_8 — draw filled triangle with antialiasing and Z buffering, 185  
 mlib\_GraphicsFillTriangle\_G\_32 — draw filled triangle with Gouraud shading, 196  
 mlib\_GraphicsFillTriangle\_G\_8 — draw filled triangle with Gouraud shading, 196  
 mlib\_GraphicsFillTriangle\_GZ\_32 — draw filled triangle with Gouraud shading and Z buffering, 198  
 mlib\_GraphicsFillTriangle\_GZ\_8 — draw filled triangle with Gouraud shading and Z buffering, 198  
 mlib\_GraphicsFillTriangle\_X\_32 — draw filled triangle in Xor mode, 218  
 mlib\_GraphicsFillTriangle\_X\_8 — draw filled triangle in Xor mode, 218  
 mlib\_GraphicsFillTriangle\_Z\_32 — draw filled triangle with Z buffering, 220  
 mlib\_GraphicsFillTriangle\_Z\_8 — draw filled triangle with Z buffering, 220  
 mlib\_GraphicsFillTriangleFanSet\_32 — draw filled triangle set where all members of the set have a common vertex, 187  
 mlib\_GraphicsFillTriangleFanSet\_8 — draw filled triangle set where all members of the set have a common vertex, 187  
 mlib\_GraphicsFillTriangleFanSet\_A\_32 — draw filled triangle set with antialiasing where all members of the set have a common vertex, 188  
 mlib\_GraphicsFillTriangleFanSet\_A\_8 — draw filled triangle set with antialiasing where all members of the set have a common vertex, 188

`mllib_GraphicsFillTriangleFanSet_AG_32` — draw filled triangle set with antialiasing and gouraud shading, where all members of the set have a common vertex, 189

`mllib_GraphicsFillTriangleFanSet_AG_8` — draw filled triangle set with antialiasing and gouraud shading, where all members of the set have a common vertex, 189

`mllib_GraphicsFillTriangleFanSet_AGZ_32` — draw filled triangle set with antialiasing, Gouraud shading, and Z buffering, where all members of the set have a common vertex, 190

`mllib_GraphicsFillTriangleFanSet_AGZ_8` — draw filled triangle set with antialiasing, Gouraud shading, and Z buffering, where all members of the set have a common vertex, 190

`mllib_GraphicsFillTriangleFanSet_AZ_32` — draw filled triangle set with antialiasing and Z buffering, where all members of the set have a common vertex, 191

`mllib_GraphicsFillTriangleFanSet_AZ_8` — draw filled triangle set with antialiasing and Z buffering, where all members of the set have a common vertex, 191

`mllib_GraphicsFillTriangleFanSet_G_32` — draw filled triangle set with Gouraud shading where all members of the set have a common vertex, 192

`mllib_GraphicsFillTriangleFanSet_G_8` — draw filled triangle set with Gouraud shading where all members of the set have a common vertex, 192

`mllib_GraphicsFillTriangleFanSet_GZ_32` — draw filled triangle set with Gouraud shading and Z buffering, where all members of the set have a common vertex, 193

`mllib_GraphicsFillTriangleFanSet_GZ_8` — draw filled triangle set with Gouraud shading and Z buffering, where all members of the set have a common vertex, 193

`mllib_GraphicsFillTriangleFanSet_X_32` — draw filled triangle set in Xor mode where all members of the set have a common vertex, 194

`mllib_GraphicsFillTriangleFanSet_X_8` — draw filled triangle set in Xor mode where all members of the set have a common vertex, 194

`mllib_GraphicsFillTriangleFanSet_Z_32` — draw filled triangle set with Z buffering where all members of the set have a common vertex, 195

`mllib_GraphicsFillTriangleFanSet_Z_8` — draw filled triangle set with Z buffering where all members of the set have a common vertex, 195

`mllib_GraphicsFillTriangleSet_32` — draw filled triangle set where each member can have different vertices, 200

`mllib_GraphicsFillTriangleSet_8` — draw filled triangle set where each member can have different vertices, 200

`mllib_GraphicsFillTriangleSet_A_32` — draw filled triangle set with antialiasing where each member can have different vertices, 201

`mllib_GraphicsFillTriangleSet_A_8` — draw filled triangle set with antialiasing where each member can have different vertices, 201

`mllib_GraphicsFillTriangleSet_AG_32` — draw filled triangle set with antialiasing and Gouraud shading, where each member can have different vertices, 202

`mllib_GraphicsFillTriangleSet_AG_8` — draw filled triangle set with antialiasing and Gouraud shading, where each member can have different vertices, 202

`mllib_GraphicsFillTriangleSet_AGZ_32` — draw filled triangle set with antialiasing, Gouraud shading, and Z buffering, where each member can have different vertices, 203

`mllib_GraphicsFillTriangleSet_AGZ_8` — draw filled triangle set with antialiasing, Gouraud shading, and Z buffering, where each member can have different vertices, 203

`mllib_GraphicsFillTriangleSet_AZ_32` — draw filled triangle set with antialiasing and Z buffering, where each member can have different vertices, 204

`mllib_GraphicsFillTriangleSet_AZ_8` — draw filled triangle set with antialiasing and Z buffering, where each member can have different vertices, 204

mlib\_GraphicsFillTriangleSet\_G\_32 — draw filled triangle set with Gouraud shading where each member can have different vertices, 205  
 mlib\_GraphicsFillTriangleSet\_G\_8 — draw filled triangle set with Gouraud shading where each member can have different vertices, 205  
 mlib\_GraphicsFillTriangleSet\_GZ\_32 — draw filled triangle set with Gouraud shading and Z buffering, where each member can have different vertices, 206  
 mlib\_GraphicsFillTriangleSet\_GZ\_8 — draw filled triangle set with Gouraud shading and Z buffering, where each member can have different vertices, 206  
 mlib\_GraphicsFillTriangleSet\_X\_32 — draw filled triangle set in Xor mode where each member can have different vertices, 207  
 mlib\_GraphicsFillTriangleSet\_X\_8 — draw filled triangle set in Xor mode where each member can have different vertices, 207  
 mlib\_GraphicsFillTriangleSet\_Z\_32 — draw filled triangle set with Z buffering where each member can have different vertices, 208  
 mlib\_GraphicsFillTriangleSet\_Z\_8 — draw filled triangle set with Z buffering where each member can have different vertices, 208  
 mlib\_GraphicsFillTriangleStripSet\_32 — draw filled triangle set where the first side of each member is common to the second side of the previous member, 209  
 mlib\_GraphicsFillTriangleStripSet\_8 — draw filled triangle set where the first side of each member is common to the second side of the previous member, 209  
 mlib\_GraphicsFillTriangleStripSet\_A\_32 — draw filled triangle set with antialiasing where the first side of each member is common to the second side of the previous member, 210  
 mlib\_GraphicsFillTriangleStripSet\_A\_8 — draw filled triangle set with antialiasing where the first side of each member is common to the second side of the previous member, 210  
 mlib\_GraphicsFillTriangleStripSet\_AG\_32 — draw filled triangle set with antialiasing and gouraud shading, where the first side of each member is common to the second side of the previous member, 211  
 mlib\_GraphicsFillTriangleStripSet\_AG\_8 — draw filled triangle set with antialiasing and gouraud shading, where the first side of each member is common to the second side of the previous member, 211  
 mlib\_GraphicsFillTriangleStripSet\_AGZ — draw filled triangle set with antialiasing, Gouraud shading, and Z buffering, where the first side of each member is common to the second side of the previous member, 212  
 mlib\_GraphicsFillTriangleStripSet\_AGZ\_32 — draw filled triangle set with antialiasing, Gouraud shading, and Z buffering, where the first side of each member is common to the second side of the previous member, 212  
 mlib\_GraphicsFillTriangleStripSet\_AGZ\_8 — draw filled triangle set with antialiasing, Gouraud shading, and Z buffering, where the first side of each member is common to the second side of the previous member, 212  
 mlib\_GraphicsFillTriangleStripSet\_AZ\_32 — draw filled triangle set with antialiasing and Z buffering, where the first side of each member is common to the second side of the previous member, 213  
 mlib\_GraphicsFillTriangleStripSet\_AZ\_8 — draw filled triangle set with antialiasing and Z buffering, where the first side of each member is common to the second side of the previous member, 213  
 mlib\_GraphicsFillTriangleStripSet\_G\_32 — draw filled triangle set with Gouraud shading where the first side of each member is common to the second side of the previous member, 214  
 mlib\_GraphicsFillTriangleStripSet\_G\_8 — draw filled triangle set with Gouraud shading where the first side of each member is common to the second side of the previous member, 214

**mllib\_GraphicsFillTriangleStripSet\_GZ\_32** — draw filled triangle set with Gouraud shading and Z buffering, where the first side of each member is common to the second side of the previous member, 215  
**mllib\_GraphicsFillTriangleStripSet\_GZ\_8** — draw filled triangle set with Gouraud shading and Z buffering, where the first side of each member is common to the second side of the previous member, 215  
**mllib\_GraphicsFillTriangleStripSet\_X\_32** — draw filled triangle set in Xor mode where the first side of each member is common to the second side of the previous member, 216  
**mllib\_GraphicsFillTriangleStripSet\_X\_8** — draw filled triangle set in Xor mode where the first side of each member is common to the second side of the previous member, 216  
**mllib\_GraphicsFillTriangleStripSet\_Z\_32** — draw filled triangle set with Z buffering where the first side of each member is common to the second side of the previous member, 217  
**mllib\_GraphicsFillTriangleStripSet\_Z\_8** — draw filled triangle set with Z buffering where the first side of each member is common to the second side of the previous member, 217  
**mllib\_GraphicsFloodFill\_32** — flood fill, 222  
**mllib\_GraphicsFloodFill\_8** — flood fill, 222  
**mllib\_ImageAbs** — computes the absolute value of the image pixels, 223  
**mllib\_ImageAbs\_Fp** — computes the absolute value of the image pixels, 224  
**mllib\_ImageAbs\_Fp\_Inp** — computes the absolute value of the image pixels, 225  
**mllib\_ImageAbs\_Inp** — computes the absolute value of the image pixels, in place, 226  
**mllib\_ImageAdd** — computes the addition of two images on a pixel-by-pixel basis, 227  
**mllib\_ImageAdd\_Fp** — computes the addition of two images on a pixel-by-pixel basis, 228  
**mllib\_ImageAdd\_Fp\_Inp** — computes the addition of two images on a pixel-by-pixel basis, 229  
**mllib\_ImageAdd\_Inp** — computes the addition of two images on a pixel-by-pixel basis, in place, 230  
**mllib\_ImageAffine** — image affine transformation, 231  
**mllib\_ImageAffine\_Fp** — image affine transformation, 233  
**mllib\_ImageAffineIndex** — affine transformation on a color indexed image, 235  
**mllib\_ImageAffineTable** — affine transformation on an image with table-driven interpolation, 237  
**mllib\_ImageAffineTable\_Fp** — affine transformation on an image with table-driven interpolation, 239  
**mllib\_ImageAffineTransform** — affine transformation on an image, checking the matrix first, 241  
**mllib\_ImageAffineTransform\_Fp** — affine transformation on an image, checking the matrix first, 243  
**mllib\_ImageAffineTransformIndex** — affine transformation on a color indexed image, checking the matrix first, 245  
**mllib\_ImageAnd** — computes the And of two images, 247  
**mllib\_ImageAnd\_Inp** — computes the And of two image, in place, 248  
**mllib\_ImageAndNot** — computes the And of the first source image and the Not of the second source image, 251  
**mllib\_ImageAndNot1\_Inp** — computes the And of the first source image and the Not of the second source image, in place, 249  
**mllib\_ImageAndNot2\_Inp** — computes the And of the first source image and the Not of the second source image, in place, 250  
**mllib\_ImageAutoCorrel** — auto-correlation of an image, 252  
**mllib\_ImageAutoCorrel\_Fp** — auto-correlation of an image, 253  
**mllib\_ImageAve** — average of two images, 254  
**mllib\_ImageAve\_Fp** — average of two images, 255  
**mllib\_ImageAve\_Fp\_Inp** — average of two images, in place, 256  
**mllib\_ImageAve\_Inp** — average of two images, in place, 257  
**mllib\_ImageBlend** — blend with an alpha image, 262

mlib\_ImageBlend\_BSRC1\_BSRC2 —  
 blending, 263  
 mlib\_ImageBlend\_BSRC1\_BSRC2\_Inp —  
 blending, in place, 266  
 mlib\_ImageBlend\_DA\_DA — blending, 263  
 mlib\_ImageBlend\_DA\_DA\_Inp — blending, in  
 place, 266  
 mlib\_ImageBlend\_DA\_DC — blending, 263  
 mlib\_ImageBlend\_DA\_DC\_Inp — blending, in  
 place, 266  
 mlib\_ImageBlend\_DA\_OMDA —  
 blending, 263  
 mlib\_ImageBlend\_DA\_OMDA\_Inp — blending,  
 in place, 266  
 mlib\_ImageBlend\_DA\_OMDC — blending, 263  
 mlib\_ImageBlend\_DA\_OMDC\_Inp — blending,  
 in place, 266  
 mlib\_ImageBlend\_DA\_OMSA — blending, 263  
 mlib\_ImageBlend\_DA\_OMSA\_Inp — blending,  
 in place, 266  
 mlib\_ImageBlend\_DA\_ONE — blending, 263  
 mlib\_ImageBlend\_DA\_ONE\_Inp — blending,  
 in place, 266  
 mlib\_ImageBlend\_DA\_SA — blending, 263  
 mlib\_ImageBlend\_DA\_SA\_Inp — blending, in  
 place, 266  
 mlib\_ImageBlend\_DA\_SAS — blending, 263  
 mlib\_ImageBlend\_DA\_SAS\_Inp — blending, in  
 place, 266  
 mlib\_ImageBlend\_DA\_ZERO — blending, 263  
 mlib\_ImageBlend\_DA\_ZERO\_Inp — blending,  
 in place, 266  
 mlib\_ImageBlend\_Fp — blend with an alpha  
 image, 273  
 mlib\_ImageBlend\_OMDA\_DA —  
 blending, 263  
 mlib\_ImageBlend\_OMDA\_DA\_Inp — blending,  
 in place, 266  
 mlib\_ImageBlend\_OMDA\_DC — blending, 263  
 mlib\_ImageBlend\_OMDA\_DC\_Inp — blending,  
 in place, 266  
 mlib\_ImageBlend\_OMDA\_OMDA —  
 blending, 263  
 mlib\_ImageBlend\_OMDA\_OMDA\_Inp —  
 blending, in place, 266  
 mlib\_ImageBlend\_OMDA\_OMDC —  
 blending, 263  
 mlib\_ImageBlend\_OMDA\_OMDC\_Inp —  
 blending, in place, 266  
 mlib\_ImageBlend\_OMDA\_OMSA —  
 blending, 263  
 mlib\_ImageBlend\_OMDA\_OMSA\_Inp —  
 blending, in place, 266  
 mlib\_ImageBlend\_OMDA\_ONE —  
 blending, 263  
 mlib\_ImageBlend\_OMDA\_ONE\_Inp —  
 blending, in place, 266  
 mlib\_ImageBlend\_OMDA\_SA — blending, 263  
 mlib\_ImageBlend\_OMDA\_SA\_Inp — blending,  
 in place, 266  
 mlib\_ImageBlend\_OMDA\_SAS —  
 blending, 263

mlib\_ImageBlend\_OMSA\_SAS\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_OMSA\_ZERO — blending, 263  
 mlib\_ImageBlend\_OMSA\_ZERO\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_OMSC\_DA — blending, 263  
 mlib\_ImageBlend\_OMSC\_DA\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_OMSC\_DC — blending, 263  
 mlib\_ImageBlend\_OMSC\_DC\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_OMSC\_OMDA — blending, 263  
 mlib\_ImageBlend\_OMSC\_OMDA\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_OMSC\_OMDC — blending, 263  
 mlib\_ImageBlend\_OMSC\_OMDC\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_OMSC\_OMSA — blending, 263  
 mlib\_ImageBlend\_OMSC\_OMSA\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_OMSC\_ONE — blending, 263  
 mlib\_ImageBlend\_OMSC\_ONE\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_OMSC\_SA — blending, 263  
 mlib\_ImageBlend\_OMSC\_SA\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_OMSC\_SAS — blending, 263  
 mlib\_ImageBlend\_OMSC\_SAS\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_OMSC\_ZERO — blending, 263  
 mlib\_ImageBlend\_OMSC\_ZERO\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_ONE\_DA — blending, 263  
 mlib\_ImageBlend\_ONE\_DA\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_ONE\_DC — blending, 263  
 mlib\_ImageBlend\_ONE\_DC\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_ONE\_OMDA — blending, 263  
 mlib\_ImageBlend\_ONE\_OMDA\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_ONE\_OMDC — blending, 263  
 mlib\_ImageBlend\_ONE\_OMDC\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_ONE\_OMSA — blending, 263  
 mlib\_ImageBlend\_ONE\_OMSA\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_ONE\_ONE — blending, 263  
 mlib\_ImageBlend\_ONE\_ONE\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_ONE\_SA — blending, 263  
 mlib\_ImageBlend\_ONE\_SA\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_ONE\_SAS — blending, 263  
 mlib\_ImageBlend\_ONE\_SAS\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_ONE\_ZERO — blending, 263  
 mlib\_ImageBlend\_ONE\_ZERO\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_SA\_DA — blending, 263  
 mlib\_ImageBlend\_SA\_DA\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_SA\_DC — blending, 263  
 mlib\_ImageBlend\_SA\_DC\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_SA\_OMDA — blending, 263  
 mlib\_ImageBlend\_SA\_OMDA\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_SA\_OMDC — blending, 263  
 mlib\_ImageBlend\_SA\_OMDC\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_SA\_OMSA — blending, 263  
 mlib\_ImageBlend\_SA\_OMSA\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_SA\_ONE — blending, 263  
 mlib\_ImageBlend\_SA\_ONE\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_SA\_SA — blending, 263  
 mlib\_ImageBlend\_SA\_SA\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_SA\_SAS — blending, 263  
 mlib\_ImageBlend\_SA\_SAS\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_SA\_ZERO — blending, 263

mlib\_ImageBlend\_SA\_ZERO\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_SC\_DA — blending, 263  
 mlib\_ImageBlend\_SC\_DA\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_SC\_DC — blending, 263  
 mlib\_ImageBlend\_SC\_DC\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_SC\_OMDA — blending, 263  
 mlib\_ImageBlend\_SC\_OMDA\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_SC\_OMDC — blending, 263  
 mlib\_ImageBlend\_SC\_OMDC\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_SC\_OMSA — blending, 263  
 mlib\_ImageBlend\_SC\_OMSA\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_SC\_ONE — blending, 263  
 mlib\_ImageBlend\_SC\_ONE\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_SC\_SA — blending, 263  
 mlib\_ImageBlend\_SC\_SA\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_SC\_SAS — blending, 263  
 mlib\_ImageBlend\_SC\_SAS\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_SC\_ZERO — blending, 263  
 mlib\_ImageBlend\_SC\_ZERO\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_ZERO\_DA — blending, 263  
 mlib\_ImageBlend\_ZERO\_DA\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_ZERO\_DC — blending, 263  
 mlib\_ImageBlend\_ZERO\_DC\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_ZERO\_OMDA — blending, 263  
 mlib\_ImageBlend\_ZERO\_OMDA\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_ZERO\_OMDC — blending, 263  
 mlib\_ImageBlend\_ZERO\_OMDC\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_ZERO\_OMSA — blending, 263  
 mlib\_ImageBlend\_ZERO\_OMSA\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_ZERO\_ONE — blending, 263  
 mlib\_ImageBlend\_ZERO\_ONE\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_ZERO\_SA — blending, 263  
 mlib\_ImageBlend\_ZERO\_SA\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_ZERO\_SAS — blending, 263  
 mlib\_ImageBlend\_ZERO\_SAS\_Inp — blending, in place, 266  
 mlib\_ImageBlend\_ZERO\_ZERO — blending, 263  
 mlib\_ImageBlend\_ZERO\_ZERO\_Inp — blending, in place, 266  
 mlib\_ImageBlend1\_Fp\_Inp — blend with an alpha image, 258  
 mlib\_ImageBlend1\_Inp — blend with an alpha image, in place, 259  
 mlib\_ImageBlend2\_Fp\_Inp — blend with an alpha image, 260  
 mlib\_ImageBlend2\_Inp — blend with an alpha image, in place, 261  
 mlib\_ImageBlendColor — blend an image and a color, 269  
 mlib\_ImageBlendColor\_Fp — blend an image and a color, 270  
 mlib\_ImageBlendColor\_Fp\_Inp — blend an image and a color, in place, 271  
 mlib\_ImageBlendColor\_Inp — blend an image and a color, in place, 272  
 mlib\_ImageBlendMulti — blend multiple images, 274  
 mlib\_ImageBlendMulti\_Fp — blend multiple images, 276  
 mlib\_ImageBlendRGBA2ARGB — image blending and channel reordering, 278  
 mlib\_ImageBlendRGBA2BGRA — image blending and channel reordering, 279  
 mlib\_ImageChannelCopy — channel copy, 280  
 mlib\_ImageChannelExtract — channel extract, 281  
 mlib\_ImageChannelInsert — channel insert, 282  
 mlib\_ImageChannelMerge — channel merge, 283  
 mlib\_ImageChannelSplit — channel split, 284  
 mlib\_ImageClear — clear, 285  
 mlib\_ImageClear\_Fp — clear, 288

**mllib\_ImageClearEdge** — sets edges of an image to a specific color, 286  
**mllib\_ImageClearEdge\_Fp** — sets edges of an image to a specific color, 287  
**mllib\_ImageColorConvert1** — color conversion using a 3x3 floating-point matrix, 289  
**mllib\_ImageColorConvert1\_Fp** — color conversion using a 3x3 floating-point matrix, 290  
**mllib\_ImageColorConvert2** — color conversion using a 3x3 floating-point matrix and a three-element offset, 291  
**mllib\_ImageColorConvert2\_Fp** — color conversion using a 3x3 floating-point matrix and a three-element offset, 292  
**mllib\_ImageColorDitherFree** — release the internal data structure for image dithering, 293  
**mllib\_ImageColorDitherInit** — initialization for image dithering, 294  
**mllib\_ImageColorErrorDiffusion3x3** — true color to indexed color conversion using error diffusion, 297  
**mllib\_ImageColorErrorDiffusionMxN** — true-color to indexed-color or grayscale to black-white conversion, using error diffusion, 298  
**mllib\_ImageColorHSL2RGB** — HSL to RGB color conversion, 300  
**mllib\_ImageColorHSL2RGB\_Fp** — HSL to RGB color conversion, 302  
**mllib\_ImageColorHSV2RGB** — HSV to RGB color conversion, 303  
**mllib\_ImageColorHSV2RGB\_Fp** — HSV to RGB color conversion, 305  
**mllib\_ImageColorOrderedDither8x8** — true color to indexed color conversion using ordered dithering, 306  
**mllib\_ImageColorOrderedDitherMxN** — true-color to indexed-color or grayscale to black-white conversion, using ordered dithering, 307  
**mllib\_ImageColorRGB2CIEMono** — RGB to monochrome conversion, 309  
**mllib\_ImageColorRGB2CIEMono\_Fp** — RGB to monochrome conversion, 310  
**mllib\_ImageColorRGB2HSL** — RGB to HSL color conversion, 311  
**mllib\_ImageColorRGB2HSL\_Fp** — RGB to HSL color conversion, 313  
**mllib\_ImageColorRGB2HSV** — RGB to HSV color conversion, 314  
**mllib\_ImageColorRGB2HSV\_Fp** — RGB to HSV color conversion, 316  
**mllib\_ImageColorRGB2Mono** — RGB to monochrome conversion, 317  
**mllib\_ImageColorRGB2Mono\_Fp** — RGB to monochrome conversion, 318  
**mllib\_ImageColorRGB2XYZ** — RGB to XYZ color conversion, 319  
**mllib\_ImageColorRGB2XYZ\_Fp** — RGB to XYZ color conversion, 320  
**mllib\_ImageColorRGB2YCC** — RGB to YCC color conversion, 321  
**mllib\_ImageColorRGB2YCC\_Fp** — RGB to YCC color conversion, 322  
**mllib\_ImageColorTrue2Index** — true color to indexed color using nearest matched LUT entries, 323  
**mllib\_ImageColorTrue2IndexFree** — releases the internal data structure for true color to indexed color conversion, 324  
**mllib\_ImageColorTrue2IndexInit** — initialization for true color to indexed color conversion, 325  
**mllib\_ImageColorXYZ2RGB** — XYZ to RGB color conversion, 327  
**mllib\_ImageColorXYZ2RGB\_Fp** — XYZ to RGB color conversion, 328  
**mllib\_ImageColorYCC2RGB** — YCC to RGB color conversion, 329  
**mllib\_ImageColorYCC2RGB\_Fp** — YCC to RGB color conversion, 330  
**mllib\_ImageComposite** — image composition, 331  
**mllib\_ImageComposite\_Inp** — image composition, in place, 333  
**mllib\_ImageConstAdd** — addition with a constant, 335  
**mllib\_ImageConstAdd\_Fp** — addition with a constant, 336  
**mllib\_ImageConstAdd\_Fp\_Inp** — addition with a constant, 337  
**mllib\_ImageConstAdd\_Inp** — addition with a constant, 338



mlib\_ImageConstAnd — And with a constant, 339  
 mlib\_ImageConstAnd\_Inp — And with a constant, 340  
 mlib\_ImageConstAndNot — AndNot with a constant, 341  
 mlib\_ImageConstAndNot\_Inp — AndNot with a constant, 342  
 mlib\_ImageConstDiv — division into a constant, 343  
 mlib\_ImageConstDiv\_Fp — division into a constant, 344  
 mlib\_ImageConstDiv\_Fp\_Inp — division into a constant, 345  
 mlib\_ImageConstDiv\_Inp — division into a constant, 346  
 mlib\_ImageConstDivShift — division into a constant, with shifting, 347  
 mlib\_ImageConstDivShift\_Inp — division into a constant, with shifting, 348  
 mlib\_ImageConstMul — multiplication with a constant, 349  
 mlib\_ImageConstMul\_Fp — multiply with a constant, 350  
 mlib\_ImageConstMul\_Fp\_Inp — multiply with a constant, 351  
 mlib\_ImageConstMul\_Inp — multiply with a constant, 352  
 mlib\_ImageConstMulShift — multiply with a constant, with shifting, 353  
 mlib\_ImageConstMulShift\_Inp — multiply with a constant, with shifting, 354  
 mlib\_ImageConstNotAnd — NotAnd with a constant, 355  
 mlib\_ImageConstNotAnd\_Inp — NotAnd with a constant, in place, 356  
 mlib\_ImageConstNotOr — NotOr with a constant, 357  
 mlib\_ImageConstNotOr\_Inp — NotOr with a constant, in place, 358  
 mlib\_ImageConstNotXor — NotXor with a constant, 359  
 mlib\_ImageConstNotXor\_Inp — NotXor with a constant, in place, 360  
 mlib\_ImageConstOr — Or with a constant, 361  
 mlib\_ImageConstOr\_Inp — Or with a constant, in place, 362  
 mlib\_ImageConstOrNot — OrNot with a constant, 363  
 mlib\_ImageConstOrNot\_Inp — OrNot with a constant, in place, 364  
 mlib\_ImageConstSub — Subtraction with a constant, 365  
 mlib\_ImageConstSub\_Fp — Subtraction with a constant, 366  
 mlib\_ImageConstSub\_Fp\_Inp — subtraction with a constant, 367  
 mlib\_ImageConstSub\_Inp — Subtraction with a constant, 368  
 mlib\_ImageConstXor — Xor with a constant, 369  
 mlib\_ImageConstXor\_Inp — Xor with a constant, in place, 370  
 mlib\_ImageConv2x2 — 2x2 convolution, 371  
 mlib\_ImageConv2x2\_Fp — 2x2 convolution, 373  
 mlib\_ImageConv2x2Index — 2x2 convolution on a color indexed image, 375  
 mlib\_ImageConv3x3 — 3x3 convolution, 377  
 mlib\_ImageConv3x3\_Fp — 3x3 convolution, 379  
 mlib\_ImageConv3x3Index — 3x3 convolution on a color indexed image, 381  
 mlib\_ImageConv4x4 — 4x4 convolution, 383  
 mlib\_ImageConv4x4\_Fp — 4x4 convolution, 385  
 mlib\_ImageConv4x4Index — 4x4 convolution on a color indexed image, 387  
 mlib\_ImageConv5x5 — 5x5 convolution, 389  
 mlib\_ImageConv5x5\_Fp — 5x5 convolution, 391  
 mlib\_ImageConv5x5Index — 5x5 convolution on a color indexed image, 393  
 mlib\_ImageConv7x7 — 7x7 convolution, 395  
 mlib\_ImageConv7x7\_Fp — 7x7 convolution, 397  
 mlib\_ImageConv7x7Index — 7x7 convolution on a color indexed image, 399  
 mlib\_ImageConvKernelConvert — convolution kernel conversion, 401  
 mlib\_ImageConvMxN — MxN convolution, 403  
 mlib\_ImageConvMxN\_Fp — MxN convolution, 405

mlib\_ImageConvMxNIndex — MxN convolution on a color indexed image, 407  
 mlib\_ImageConvolveMxN — MxN convolution, with kernel analysis for taking advantage of special cases, 409  
 mlib\_ImageConvolveMxN\_Fp — MxN convolution, with kernel analysis for taking advantage of special cases, 411  
 mlib\_ImageCopy — image copy, 413  
 mlib\_ImageCopyArea — copy an area, 414  
 mlib\_ImageCopyMask — copy with mask, 415  
 mlib\_ImageCopyMask\_Fp — copy with mask, floating-point, 416  
 mlib\_ImageCopySubimage — copy subimage, 417  
 mlib\_ImageCreate — image creation, 418  
 mlib\_ImageCreateStruct — image structure creation, 419  
 mlib\_ImageCreateSubimage — subimage creation, 420  
 mlib\_ImageCrossCorrel — cross correlation, 421  
 mlib\_ImageCrossCorrel\_Fp — cross correlation, 422  
 mlib\_ImageDataTypeConvert — data type conversion, 423  
 mlib\_ImageDelete — image delete, 426  
 mlib\_ImageDilate4 — four neighbor dilate, 427  
 mlib\_ImageDilate4\_Fp — four neighbor dilate, 428  
 mlib\_ImageDilate8 — eight neighbor dilate, 429  
 mlib\_ImageDilate8\_Fp — eight neighbor dilate, 430  
 mlib\_ImageDiv\_Fp — division, 441  
 mlib\_ImageDiv1\_Fp\_Inp — division, in place, 431  
 mlib\_ImageDiv2\_Fp\_Inp — division, in place, 432  
 mlib\_ImageDivAlpha — alpha channel division, 433  
 mlib\_ImageDivAlpha\_Fp — alpha channel division, 435  
 mlib\_ImageDivAlpha\_Fp\_Inp — alpha channel division, in place, 436  
 mlib\_ImageDivAlpha\_Inp — alpha channel division, in place, 437  
 mlib\_ImageDivConstShift — division by a constant, with shifting, 439  
 mlib\_ImageDivConstShift\_Inp — division by a constant, with shifting, 440  
 mlib\_ImageDivShift — division with shifting, 444  
 mlib\_ImageDivShift1\_Inp — division with shifting, in place, 442  
 mlib\_ImageDivShift2\_Inp — division with shifting, in place, 443  
 mlib\_ImageErode4 — four neighbor erode, 445  
 mlib\_ImageErode4\_Fp — four neighbor erode, 446  
 mlib\_ImageErode8 — eight neighbor erode, 447  
 mlib\_ImageErode8\_Fp — eight neighbor erode, 448  
 mlib\_ImageExp — computes the exponent of the image pixels, 449  
 mlib\_ImageExp\_Fp — computes the exponent of the image pixels, 450  
 mlib\_ImageExp\_Fp\_Inp — computes the exponent of the image pixels, 451  
 mlib\_ImageExp\_Inp — computes the exponent of the image pixels, 452  
 mlib\_ImageExtrema2 — image extrema, 453  
 mlib\_ImageExtrema2\_Fp — image extrema, 453  
 mlib\_ImageExtremaLocations — image extrema and their locations, 455  
 mlib\_ImageExtremaLocations\_Fp — image extrema and their locations, 455  
 mlib\_ImageFilteredSubsample — antialias filters and subsamples an image, 458  
 mlib\_ImageFilteredSubsample\_Fp — antialias filters and subsamples an image, 458  
 mlib\_ImageFlipAntiDiag — anti-diagonal flip, 461  
 mlib\_ImageFlipAntiDiag\_Fp — anti-diagonal flip, 462  
 mlib\_ImageFlipMainDiag — main diagonal flip, 463  
 mlib\_ImageFlipMainDiag\_Fp — main diagonal flip, 464  
 mlib\_ImageFlipX — X-axis flip, 465  
 mlib\_ImageFlipX\_Fp — X-axis flip, 466  
 mlib\_ImageFlipY — Y-axis flip, 467  
 mlib\_ImageFlipY\_Fp — Y-axis flip, 468

mlib\_ImageFourierTransform — Fourier transform, 469  
 mlib\_ImageGetBitOffset — get bitoffset, 471  
 mlib\_ImageGetChannels — get channels, 472  
 mlib\_ImageGetData — get data, 473  
 mlib\_ImageGetFlags — get flags, 474  
 mlib\_ImageGetFormat — get format, 475  
 mlib\_ImageGetHeight — get height, 476  
 mlib\_ImageGetPaddings — get paddings, 477  
 mlib\_ImageGetStride — get stride, 478  
 mlib\_ImageGetType — get type, 479  
 mlib\_ImageGetWidth — get width, 480  
 mlib\_ImageGradient3x3 — 3x3 gradient filter, 481  
 mlib\_ImageGradient3x3\_Fp — 3x3 gradient filter, 483  
 mlib\_ImageGradientMxN — MxN gradient filter, 485  
 mlib\_ImageGradientMxN\_Fp — MxN gradient filter, 487  
 mlib\_ImageGridWarp — grid-based image warp, 489  
 mlib\_ImageGridWarp\_Fp — grid-based image warp, 492  
 mlib\_ImageGridWarpTable — grid-based image warp with table-driven interpolation, 495  
 mlib\_ImageGridWarpTable\_Fp — grid-based image warp with table-driven interpolation, 498  
 mlib\_ImageHistogram — histogram, 503  
 mlib\_ImageHistogram2 — histogram, 501  
 mlib\_ImageInterpTableCreate — creates an interpolation table, 504  
 mlib\_ImageInterpTableDelete — deletes an interpolation table, 506  
 mlib\_ImageInvert — invert, 507  
 mlib\_ImageInvert\_Fp — invert, 508  
 mlib\_ImageInvert\_Fp\_Inp — invert, 509  
 mlib\_ImageInvert\_Inp — invert in place, 510  
 mlib\_ImageIsNotAligned2 — image query, two-byte aligned, 511  
 mlib\_ImageIsNotAligned4 — image query, four-byte aligned, 512  
 mlib\_ImageIsNotAligned64 — image query, 64-byte aligned, 513  
 mlib\_ImageIsNotAligned8 — image query, eight-byte aligned, 514  
 mlib\_ImageIsNotHeight2X — image query, 2X height, 515  
 mlib\_ImageIsNotHeight4X — image query, 4X height, 516  
 mlib\_ImageIsNotHeight8X — image query, 8X height, 517  
 mlib\_ImageIsNotOneDvector — image query, 1D vector, 518  
 mlib\_ImageIsNotStride8X — image query, 8X stride, 519  
 mlib\_ImageIsNotWidth2X — image query, 2X width, 520  
 mlib\_ImageIsNotWidth4X — image query, 4X width, 521  
 mlib\_ImageIsNotWidth8X — image query, 8X width, 522  
 mlib\_ImageIsUserAllocated — image query, user-allocated, 523  
 mlib\_ImageLog — computes the natural logarithm of the image pixels, 524  
 mlib\_ImageLog\_Fp — computes the natural logarithm of the image pixels, 525  
 mlib\_ImageLog\_Fp\_Inp — computes the natural logarithm of the image pixels, 526  
 mlib\_ImageLog\_Inp — computes the natural logarithm of the image pixels, in place, 527  
 mlib\_ImageLookUp — table lookup, 530  
 mlib\_ImageLookUp\_Inp — table lookup, in place, 532  
 mlib\_ImageLookUp2 — table lookup, 528  
 mlib\_ImageLookUpMask — table lookup with mask, 533  
 mlib\_ImageMax — two image maximum, 535  
 mlib\_ImageMax\_Fp — two image maximum, 548  
 mlib\_ImageMax\_Fp\_Inp — two image maximum, 549  
 mlib\_ImageMax\_Inp — two image maximum, 552  
 mlib\_ImageMaxFilter3x3 — 3x3 maximum filter, 536  
 mlib\_ImageMaxFilter3x3\_Fp — 3x3 maximum filter, 538  
 mlib\_ImageMaxFilter5x5 — 5x5 Max Filter, 540  
 mlib\_ImageMaxFilter5x5\_Fp — 5x5 Max Filter, 542  
 mlib\_ImageMaxFilter7x7 — 7x7 Max Filter, 544

mlib\_ImageMaxFilter7x7\_Fp — 7x7 Max Filter, 546  
 mlib\_ImageMaximum — image maximum, 550  
 mlib\_ImageMaximum\_Fp — image maximum, 551  
 mlib\_ImageMean — image mean, 553  
 mlib\_ImageMean\_Fp — image mean, 554  
 mlib\_ImageMedianFilter3x3 — 3x3 median filter, 555  
 mlib\_ImageMedianFilter3x3\_Fp — 3x3 median filter, 557  
 mlib\_ImageMedianFilter3x3\_US — 3x3 median filter, unsigned, 559  
 mlib\_ImageMedianFilter5x5 — 5x5 median filter, 561  
 mlib\_ImageMedianFilter5x5\_Fp — 5x5 median filter, 563  
 mlib\_ImageMedianFilter5x5\_US — 5x5 median filter, unsigned, 565  
 mlib\_ImageMedianFilter7x7 — 7x7 median filter, 567  
 mlib\_ImageMedianFilter7x7\_Fp — 7x7 median filter, 569  
 mlib\_ImageMedianFilter7x7\_US — 7x7 median filter, unsigned, 571  
 mlib\_ImageMedianFilterMxN — MxN median filter, 573  
 mlib\_ImageMedianFilterMxN\_Fp — MxN median filter, 575  
 mlib\_ImageMedianFilterMxN\_US — MxN median filter, unsigned, 577  
 mlib\_ImageMin — two-image minimum, 579  
 mlib\_ImageMin\_Fp — two-image minimum, 592  
 mlib\_ImageMin\_Fp\_Inp — two-image minimum, 593  
 mlib\_ImageMin\_Inp — two-image minimum, 596  
 mlib\_ImageMinFilter3x3 — 3x3 Min Filter, 580  
 mlib\_ImageMinFilter3x3\_Fp — 3x3 Min Filter, floating point, 582  
 mlib\_ImageMinFilter5x5 — 5x5 Min Filter, 584  
 mlib\_ImageMinFilter5x5\_Fp — 5x5 Min Filter, floating point, 586  
 mlib\_ImageMinFilter7x7 — 7x7 Min Filter, 588  
 mlib\_ImageMinFilter7x7\_Fp — 7x7 Min Filter, floating point, 590  
 mlib\_ImageMinimum — image minimum, 594  
 mlib\_ImageMinimum\_Fp — image minimum, 595  
 mlib\_ImageMoment2 — second moment, 597  
 mlib\_ImageMoment2\_Fp — second moment, 598  
 mlib\_ImageMul\_Fp — computes the multiplication of two images on a pixel-by-pixel basis, 603  
 mlib\_ImageMul\_Fp\_Inp — computes the multiplication of two images on a pixel-by-pixel basis, 604  
 mlib\_ImageMulAlpha — alpha channel multiplication, 599  
 mlib\_ImageMulAlpha\_Fp — alpha channel multiplication, 600  
 mlib\_ImageMulAlpha\_Fp\_Inp — alpha channel multiplication, 601  
 mlib\_ImageMulAlpha\_Inp — alpha channel multiplication, in place, 602  
 mlib\_ImageMulShift — multiplication, 605  
 mlib\_ImageMulShift\_Inp — multiplication, in place, 606  
 mlib\_ImageNot — Not, 607  
 mlib\_ImageNot\_Inp — Not, 610  
 mlib\_ImageNotAnd — NotAnd, 608  
 mlib\_ImageNotAnd\_Inp — NotAnd, in place, 609  
 mlib\_ImageNotOr — NotOr, 611  
 mlib\_ImageNotOr\_Inp — NotOr, in place, 612  
 mlib\_ImageNotXor — NotXor, 613  
 mlib\_ImageNotXor\_Inp — NotXor, in place, 614  
 mlib\_ImageOr — Or, 615  
 mlib\_ImageOr\_Inp — Or, in place, 616  
 mlib\_ImageOrNot — OrNot, 619  
 mlib\_ImageOrNot1\_Inp — OrNot, in place, 617  
 mlib\_ImageOrNot2\_Inp — OrNot, in place, 618  
 mlib\_ImagePolynomialWarp — polynomial-based image warp, 620  
 mlib\_ImagePolynomialWarp\_Fp — polynomial-based image warp, 623  
 mlib\_ImagePolynomialWarpTable — polynomial-based image warp with table-driven interpolation, 626  
 mlib\_ImagePolynomialWarpTable\_Fp — polynomial-based image warp with table-driven interpolation, 629  
 mlib\_ImageRankFilter3x3 — 3x3 rank filter, 632

mlib\_ImageRankFilter3x3\_Fp — 3x3 rank filter, 634  
 mlib\_ImageRankFilter3x3\_US — 3x3 rank filter, unsigned, 636  
 mlib\_ImageRankFilter5x5 — 5x5 rank filter, 638  
 mlib\_ImageRankFilter5x5\_Fp — 5x5 rank filter, 640  
 mlib\_ImageRankFilter5x5\_US — 5x5 rank filter, unsigned, 642  
 mlib\_ImageRankFilter7x7 — 7x7 rank filter, 644  
 mlib\_ImageRankFilter7x7\_Fp — 7x7 rank filter, 646  
 mlib\_ImageRankFilter7x7\_US — 7x7 rank filter, unsigned, 648  
 mlib\_ImageRankFilterMxN — MxN rank filter, 650  
 mlib\_ImageRankFilterMxN\_Fp — MxN rank filter, 652  
 mlib\_ImageRankFilterMxN\_US — MxN rank filter, unsigned, 654  
 mlib\_ImageReformat — image data buffer reformat, 656  
 mlib\_ImageReplaceColor — replace a color in an image, 659  
 mlib\_ImageReplaceColor\_Fp — replace a color in an image, 660  
 mlib\_ImageReplaceColor\_Fp\_Inp — replace a color in an image, in place, 661  
 mlib\_ImageReplaceColor\_Inp — replace a color in an image, in place, 662  
 mlib\_ImageRotate — rotate image, 667  
 mlib\_ImageRotate\_Fp — rotate image, 671  
 mlib\_ImageRotate180 — rotate an image by 180 degrees, 663  
 mlib\_ImageRotate180\_Fp — rotate an image by 180 degrees, 664  
 mlib\_ImageRotate270 — rotate an image by 270 degrees, 665  
 mlib\_ImageRotate270\_Fp — rotate an image by 270 degrees, 666  
 mlib\_ImageRotate90 — rotate an image by 90 degrees, 669  
 mlib\_ImageRotate90\_Fp — rotate an image by 90 degrees, 670  
 mlib\_ImageRotateIndex — rotate color-indexed image, 673  
 mlib\_ImageScalarBlend — image blending with scalar, 675  
 mlib\_ImageScalarBlend\_Fp — image blending with scalar, 676  
 mlib\_ImageScalarBlend\_Fp\_Inp — image blending with scalar, 677  
 mlib\_ImageScalarBlend\_Inp — image blending with scalar, in place, 678  
 mlib\_ImageScale — linear scaling, 682  
 mlib\_ImageScale\_Fp — linear scaling, 684  
 mlib\_ImageScale\_Fp\_Inp — linear scaling, in place, 686  
 mlib\_ImageScale\_Inp — linear scaling, in place, 687  
 mlib\_ImageScale2 — linear scaling, 679  
 mlib\_ImageScale2\_Inp — linear scaling, in place, 681  
 mlib\_ImageSConv3x3 — separable 3x3 convolution, 688  
 mlib\_ImageSConv3x3\_Fp — separable 3x3 convolution, 690  
 mlib\_ImageSConv5x5 — separable 5x5 convolution, 692  
 mlib\_ImageSConv5x5\_Fp — separable 5x5 convolution, 694  
 mlib\_ImageSConv7x7 — separable 7x7 convolution, 696  
 mlib\_ImageSConv7x7\_Fp — separable 7x7 convolution, 698  
 mlib\_ImageSConvKernelConvert — kernel conversion for separable convolution, 700  
 mlib\_ImageSetFormat — set format, 701  
 mlib\_ImageSetPaddings — set paddings, 702  
 mlib\_ImageSobel — Sobel filter, 704  
 mlib\_ImageSobel\_Fp — Sobel filter, 704  
 mlib\_ImageSqr\_Fp — square, 706  
 mlib\_ImageSqr\_Fp\_Inp — square, in place, 707  
 mlib\_ImageSqrShift — square with shifting, 708  
 mlib\_ImageSqrShift\_Inp — square with shifting, in place, 709  
 mlib\_ImageStdDev — image standard deviation, 710  
 mlib\_ImageStdDev\_Fp — image standard deviation, 711  
 mlib\_ImageSub — subtraction, 716  
 mlib\_ImageSub\_Fp — subtraction, 717  
 mlib\_ImageSub1\_Fp\_Inp — subtraction, in place, 712

mlib\_ImageSub1\_Inp — subtraction, in place, 713  
 mlib\_ImageSub2\_Fp\_Inp — subtraction, in place, 714  
 mlib\_ImageSub2\_Inp — subtraction, in place, 715  
 mlib\_ImageSubsampleAverage — subsamples an image with a box filter, 718  
 mlib\_ImageSubsampleAverage\_Fp — subsamples an image with a box filter, 718  
 mlib\_ImageSubsampleBinaryToGray — subsamples a binary image and converts it to a grayscale image, 720  
 mlib\_ImageTestFlags — test flags, 722  
 mlib\_ImageThresh1 — image thresholding, 723  
 mlib\_ImageThresh1\_Fp — image thresholding, 725  
 mlib\_ImageThresh1\_Fp\_Inp — image thresholding, 727  
 mlib\_ImageThresh1\_Inp — image thresholding, 728  
 mlib\_ImageThresh2 — image thresholding, 729  
 mlib\_ImageThresh2\_Fp — image thresholding, 730  
 mlib\_ImageThresh2\_Fp\_Inp — image thresholding, 731  
 mlib\_ImageThresh2\_Inp — image thresholding, 732  
 mlib\_ImageThresh3 — image thresholding, 733  
 mlib\_ImageThresh3\_Fp — image thresholding, 734  
 mlib\_ImageThresh3\_Fp\_Inp — image thresholding, 735  
 mlib\_ImageThresh3\_Inp — image thresholding, 736  
 mlib\_ImageThresh4 — image thresholding, 737  
 mlib\_ImageThresh4\_Fp — image thresholding, 739  
 mlib\_ImageThresh4\_Fp\_Inp — image thresholding, 741  
 mlib\_ImageThresh4\_Inp — image thresholding, 743  
 mlib\_ImageThresh5 — image thresholding, 745  
 mlib\_ImageThresh5\_Fp — image thresholding, 747  
 mlib\_ImageThresh5\_Fp\_Inp — image thresholding, 749  
 mlib\_ImageThresh5\_Inp — image thresholding, 750  
 mlib\_ImageXor — Xor, 751  
 mlib\_ImageXor\_Inp — Xor, in place, 752  
 mlib\_ImageXProj — image X projection, 753  
 mlib\_ImageXProj\_Fp — image X projection, 754  
 mlib\_ImageYProj — image Y projection, 755  
 mlib\_ImageYProj\_Fp — image Y projection, 756  
 mlib\_ImageZoom — zoom, 757  
 mlib\_ImageZoom\_Fp — zoom, 761  
 mlib\_ImageZoomBlend — image scaling with alpha blending, 759  
 mlib\_ImageZoomIn2X — 2X zoom, 763  
 mlib\_ImageZoomIn2X\_Fp — 2X zoom, 764  
 mlib\_ImageZoomIn2XIndex — 2X zoom on color-indexed image, 766  
 mlib\_ImageZoomIndex — zoom on color-indexed image, 768  
 mlib\_ImageZoomOut2X — 0.5X zoom, 770  
 mlib\_ImageZoomOut2X\_Fp — 0.5X zoom, 771  
 mlib\_ImageZoomOut2XIndex — 0.5X zoom, 773  
 mlib\_ImageZoomTranslate — zoom, with translation, 775  
 mlib\_ImageZoomTranslate\_Fp — zoom, with translation, 780  
 mlib\_ImageZoomTranslateBlend — image scaling with alpha blending, 777  
 mlib\_ImageZoomTranslateTable — zoom, with translation, with table-driven interpolation, 782  
 mlib\_ImageZoomTranslateTable\_Fp — zoom, with translation, with table-driven interpolation, 787  
 mlib\_ImageZoomTranslateTableBlend — image scaling using interpolation table, combined with alpha blending, 784  
 mlib\_ImageZoomTranslateToGray — zoom, with translation, and convert to grayscale, 789  
 mlib\_malloc — allocate a block of bytes, 792  
 mlib\_MatrixAdd\_S16\_Mod — matrix addition, in place, 798  
 mlib\_MatrixAdd\_S16\_S16\_Mod — matrix addition, 800

mlib\_MatrixAdd\_S16\_S16\_Sat — matrix addition, 800  
 mlib\_MatrixAdd\_S16\_S8\_Mod — matrix addition, 800  
 mlib\_MatrixAdd\_S16\_S8\_Sat — matrix addition, 800  
 mlib\_MatrixAdd\_S16\_Sat — matrix addition, in place, 798  
 mlib\_MatrixAdd\_S16\_U8\_Mod — matrix addition, 800  
 mlib\_MatrixAdd\_S16\_U8\_Sat — matrix addition, 800  
 mlib\_MatrixAdd\_S16C\_Mod — matrix addition, in place, 798  
 mlib\_MatrixAdd\_S16C\_S16C\_Mod — matrix addition, 800  
 mlib\_MatrixAdd\_S16C\_S16C\_Sat — matrix addition, 800  
 mlib\_MatrixAdd\_S16C\_S8C\_Mod — matrix addition, 800  
 mlib\_MatrixAdd\_S16C\_S8C\_Sat — matrix addition, 800  
 mlib\_MatrixAdd\_S16C\_Sat — matrix addition, in place, 798  
 mlib\_MatrixAdd\_S16C\_U8C\_Mod — matrix addition, 800  
 mlib\_MatrixAdd\_S16C\_U8C\_Sat — matrix addition, 800  
 mlib\_MatrixAdd\_S32\_Mod — matrix addition, in place, 798  
 mlib\_MatrixAdd\_S32\_S16\_Mod — matrix addition, 800  
 mlib\_MatrixAdd\_S32\_S16\_Sat — matrix addition, 800  
 mlib\_MatrixAdd\_S32\_S32\_Mod — matrix addition, 800  
 mlib\_MatrixAdd\_S32\_S32\_Sat — matrix addition, 800  
 mlib\_MatrixAdd\_S32\_Sat — matrix addition, in place, 798  
 mlib\_MatrixAdd\_S32C\_Mod — matrix addition, in place, 798  
 mlib\_MatrixAdd\_S32C\_S16C\_Mod — matrix addition, 800  
 mlib\_MatrixAdd\_S32C\_S16C\_Sat — matrix addition, 800  
 mlib\_MatrixAdd\_S32C\_S32C\_Mod — matrix addition, 800  
 mlib\_MatrixAdd\_S32C\_S32C\_Sat — matrix addition, 800  
 mlib\_MatrixAdd\_S32C\_Sat — matrix addition, in place, 798  
 mlib\_MatrixAdd\_S8\_Mod — matrix addition, in place, 798  
 mlib\_MatrixAdd\_S8\_S8\_Mod — matrix addition, 800  
 mlib\_MatrixAdd\_S8\_S8\_Sat — matrix addition, 800  
 mlib\_MatrixAdd\_S8\_Sat — matrix addition, in place, 798  
 mlib\_MatrixAdd\_S8C\_Mod — matrix addition, in place, 798  
 mlib\_MatrixAdd\_S8C\_S8C\_Mod — matrix addition, 800  
 mlib\_MatrixAdd\_S8C\_S8C\_Sat — matrix addition, 800  
 mlib\_MatrixAdd\_S8C\_Sat — matrix addition, in place, 798  
 mlib\_MatrixAdd\_U8\_Mod — matrix addition, in place, 798  
 mlib\_MatrixAdd\_U8\_Sat — matrix addition, in place, 798  
 mlib\_MatrixAdd\_U8\_U8\_Mod — matrix addition, 800  
 mlib\_MatrixAdd\_U8\_U8\_Sat — matrix addition, 800  
 mlib\_MatrixAdd\_U8C\_Mod — matrix addition, in place, 798  
 mlib\_MatrixAdd\_U8C\_Sat — matrix addition, in place, 798  
 mlib\_MatrixAdd\_U8C\_U8C\_Mod — matrix addition, 800  
 mlib\_MatrixAdd\_U8C\_U8C\_Sat — matrix addition, 800  
 mlib\_MatrixAddS\_S16\_Mod — matrix addition to scalar, in place, 793  
 mlib\_MatrixAddS\_S16\_S16\_Mod — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_S16\_S16\_Sat — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_S16\_S8\_Mod — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_S16\_S8\_Sat — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_S16\_Sat — matrix addition to scalar, in place, 793

mlib\_MatrixAddS\_S16\_U8\_Mod — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_S16\_U8\_Sat — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_S16C\_Mod — matrix addition to scalar, in place, 793  
 mlib\_MatrixAddS\_S16C\_S16C\_Mod — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_S16C\_S16C\_Sat — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_S16C\_S8C\_Mod — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_S16C\_S8C\_Sat — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_S16C\_Sat — matrix addition to scalar, in place, 793  
 mlib\_MatrixAddS\_S16C\_U8C\_Mod — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_S16C\_U8C\_Sat — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_S32\_Mod — matrix addition to scalar, in place, 793  
 mlib\_MatrixAddS\_S32\_S16\_Mod — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_S32\_S16\_Sat — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_S32\_S32\_Mod — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_S32\_S32\_Sat — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_S32\_Sat — matrix addition to scalar, in place, 793  
 mlib\_MatrixAddS\_S32C\_Mod — matrix addition to scalar, in place, 793  
 mlib\_MatrixAddS\_S32C\_S16C\_Mod — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_S32C\_S16C\_Sat — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_S32C\_S32C\_Mod — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_S32C\_S32C\_Sat — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_S32C\_Sat — matrix addition to scalar, in place, 793  
 mlib\_MatrixAddS\_S8\_Mod — matrix addition to scalar, in place, 793  
 mlib\_MatrixAddS\_S8\_S8\_Mod — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_S8\_S8\_Sat — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_S8\_Sat — matrix addition to scalar, in place, 793  
 mlib\_MatrixAddS\_S8C\_Mod — matrix addition to scalar, in place, 793  
 mlib\_MatrixAddS\_S8C\_S8C\_Mod — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_S8C\_S8C\_Sat — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_S8C\_Sat — matrix addition to scalar, in place, 793  
 mlib\_MatrixAddS\_U8\_Mod — matrix addition to scalar, in place, 793  
 mlib\_MatrixAddS\_U8\_Sat — matrix addition to scalar, in place, 793  
 mlib\_MatrixAddS\_U8\_U8\_Mod — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_U8\_U8\_Sat — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_U8C\_Mod — matrix addition to scalar, in place, 793  
 mlib\_MatrixAddS\_U8C\_Sat — matrix addition to scalar, in place, 793  
 mlib\_MatrixAddS\_U8C\_U8C\_Mod — matrix addition to scalar, 795  
 mlib\_MatrixAddS\_U8C\_U8C\_Sat — matrix addition to scalar, 795  
 mlib\_MatrixMaximum\_D64 — find the maximum value in a matrix, 804  
 mlib\_MatrixMaximum\_F32 — find the maximum value in a matrix, 804  
 mlib\_MatrixMaximum\_S16 — find the maximum value in a matrix, 804  
 mlib\_MatrixMaximum\_S32 — find the maximum value in a matrix, 804  
 mlib\_MatrixMaximum\_S8 — find the maximum value in a matrix, 804  
 mlib\_MatrixMaximum\_U8 — find the maximum value in a matrix, 804  
 mlib\_MatrixMaximumMag\_D64C — find the first element with the maximum magnitude in a matrix, 803  
 mlib\_MatrixMaximumMag\_F32C — find the first element with the maximum magnitude in a matrix, 803



mlib\_MatrixMaximumMag\_S16C — find the first element with the maximum magnitude in a matrix, 803  
 mlib\_MatrixMaximumMag\_S32C — find the first element with the maximum magnitude in a matrix, 803  
 mlib\_MatrixMaximumMag\_S8C — find the first element with the maximum magnitude in a matrix, 803  
 mlib\_MatrixMaximumMag\_U8C — find the first element with the maximum magnitude in a matrix, 803  
 mlib\_MatrixMinimum\_D64 — find the minimum value in a matrix, 806  
 mlib\_MatrixMinimum\_F32 — find the minimum value in a matrix, 806  
 mlib\_MatrixMinimum\_S16 — find the minimum value in a matrix, 806  
 mlib\_MatrixMinimum\_S32 — find the minimum value in a matrix, 806  
 mlib\_MatrixMinimum\_S8 — find the minimum value in a matrix, 806  
 mlib\_MatrixMinimum\_U8 — find the minimum value in a matrix, 806  
 mlib\_MatrixMinimumMag\_D64C — find the first element with the minimum magnitude in a matrix, 805  
 mlib\_MatrixMinimumMag\_F32C — find the first element with the minimum magnitude in a matrix, 805  
 mlib\_MatrixMinimumMag\_S16C — find the first element with the minimum magnitude in a matrix, 805  
 mlib\_MatrixMinimumMag\_S32C — find the first element with the minimum magnitude in a matrix, 805  
 mlib\_MatrixMinimumMag\_S8C — find the first element with the minimum magnitude in a matrix, 805  
 mlib\_MatrixMinimumMag\_U8C — find the first element with the minimum magnitude in a matrix, 805  
 mlib\_MatrixMul\_S16\_S16\_Mod — matrix multiplication, 819  
 mlib\_MatrixMul\_S16\_S16\_Sat — matrix multiplication, 819  
 mlib\_MatrixMul\_S16\_S8\_Mod — matrix multiplication, 819  
 mlib\_MatrixMul\_S16\_S8\_Sat — matrix multiplication, 819  
 mlib\_MatrixMul\_S16C\_S16C\_Mod — matrix multiplication, 819  
 mlib\_MatrixMul\_S16C\_S16C\_Sat — matrix multiplication, 819  
 mlib\_MatrixMul\_S16C\_S8C\_Mod — matrix multiplication, 819  
 mlib\_MatrixMul\_S16C\_S8C\_Sat — matrix multiplication, 819  
 mlib\_MatrixMul\_S16C\_U8C\_Mod — matrix multiplication, 819  
 mlib\_MatrixMul\_S16C\_U8C\_Sat — matrix multiplication, 819  
 mlib\_MatrixMul\_S32\_S16\_Mod — matrix multiplication, 819  
 mlib\_MatrixMul\_S32\_S16\_Sat — matrix multiplication, 819  
 mlib\_MatrixMul\_S32\_S32\_Mod — matrix multiplication, 819  
 mlib\_MatrixMul\_S32\_S32\_Sat — matrix multiplication, 819  
 mlib\_MatrixMul\_S32C\_S16C\_Mod — matrix multiplication, 819  
 mlib\_MatrixMul\_S32C\_S16C\_Sat — matrix multiplication, 819  
 mlib\_MatrixMul\_S32C\_S32C\_Mod — matrix multiplication, 819  
 mlib\_MatrixMul\_S32C\_S32C\_Sat — matrix multiplication, 819  
 mlib\_MatrixMul\_S8\_S8\_Mod — matrix multiplication, 819  
 mlib\_MatrixMul\_S8\_S8\_Sat — matrix multiplication, 819  
 mlib\_MatrixMul\_S8C\_S8C\_Mod — matrix multiplication, 819  
 mlib\_MatrixMul\_S8C\_S8C\_Sat — matrix multiplication, 819  
 mlib\_MatrixMul\_U8\_U8\_Mod — matrix multiplication, 819  
 mlib\_MatrixMul\_U8\_U8\_Sat — matrix multiplication, 819  
 mlib\_MatrixMul\_U8C\_U8C\_Mod — matrix multiplication, 819

mlib\_MatrixMul\_U8C\_U8C\_Sat — matrix multiplication, 819  
 mlib\_MatrixMulS\_S16\_Mod — matrix multiplication by scalar, in place, 814  
 mlib\_MatrixMulS\_S16\_S16\_Mod — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_S16\_S16\_Sat — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_S16\_S8\_Mod — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_S16\_S8\_Sat — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_S16\_Sat — matrix multiplication by scalar, in place, 814  
 mlib\_MatrixMulS\_S16\_U8\_Mod — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_S16\_U8\_Sat — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_S16C\_Mod — matrix multiplication by scalar, in place, 814  
 mlib\_MatrixMulS\_S16C\_S16C\_Mod — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_S16C\_S16C\_Sat — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_S16C\_S8C\_Mod — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_S16C\_S8C\_Sat — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_S16C\_Sat — matrix multiplication by scalar, in place, 814  
 mlib\_MatrixMulS\_S16C\_U8C\_Mod — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_S16C\_U8C\_Sat — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_S32\_Mod — matrix multiplication by scalar, in place, 814  
 mlib\_MatrixMulS\_S32\_S16\_Mod — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_S32\_S16\_Sat — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_S32\_S32\_Mod — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_S32\_S32\_Sat — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_S32\_Sat — matrix multiplication by scalar, in place, 814  
 mlib\_MatrixMulS\_S32C\_Mod — matrix multiplication by scalar, in place, 814  
 mlib\_MatrixMulS\_S32C\_S16C\_Mod — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_S32C\_S16C\_Sat — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_S32C\_S32C\_Mod — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_S32C\_S32C\_Sat — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_S32C\_Sat — matrix multiplication by scalar, in place, 814  
 mlib\_MatrixMulS\_S8\_Mod — matrix multiplication by scalar, in place, 814  
 mlib\_MatrixMulS\_S8\_S8\_Mod — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_S8\_S8\_Sat — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_S8\_Sat — matrix multiplication by scalar, in place, 814  
 mlib\_MatrixMulS\_S8C\_Mod — matrix multiplication by scalar, in place, 814  
 mlib\_MatrixMulS\_S8C\_S8C\_Mod — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_S8C\_S8C\_Sat — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_S8C\_Sat — matrix multiplication by scalar, in place, 814  
 mlib\_MatrixMulS\_U8\_Mod — matrix multiplication by scalar, in place, 814  
 mlib\_MatrixMulS\_U8\_Sat — matrix multiplication by scalar, in place, 814  
 mlib\_MatrixMulS\_U8\_U8\_Mod — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_U8\_U8\_Sat — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_U8C\_Mod — matrix multiplication by scalar, in place, 814  
 mlib\_MatrixMulS\_U8C\_Sat — matrix multiplication by scalar, in place, 814  
 mlib\_MatrixMulS\_U8C\_U8C\_Mod — matrix multiplication by scalar, 816  
 mlib\_MatrixMulS\_U8C\_U8C\_Sat — matrix multiplication by scalar, 816  
 mlib\_MatrixMulShift\_S16\_S16\_Mod — matrix multiplication plus shifting, 807  
 mlib\_MatrixMulShift\_S16\_S16\_Sat — matrix multiplication plus shifting, 807  
 mlib\_MatrixMulShift\_S16C\_S16C\_Mod — matrix multiplication plus shifting, 807

mlib\_MatrixMulShift\_S16C\_S16C\_Sat — matrix multiplication plus shifting, 807  
 mlib\_MatrixMulSShift\_S16\_Mod — matrix multiplication by scalar plus shifting, in place, 809  
 mlib\_MatrixMulSShift\_S16\_S16\_Mod — matrix multiplication by scalar plus shifting, 811  
 mlib\_MatrixMulSShift\_S16\_S16\_Sat — matrix multiplication by scalar plus shifting, 811  
 mlib\_MatrixMulSShift\_S16\_Sat — matrix multiplication by scalar plus shifting, in place, 809  
 mlib\_MatrixMulSShift\_S16C\_Mod — matrix multiplication by scalar plus shifting, in place, 809  
 mlib\_MatrixMulSShift\_S16C\_S16C\_Mod — matrix multiplication by scalar plus shifting, 811  
 mlib\_MatrixMulSShift\_S16C\_S16C\_Sat — matrix multiplication by scalar plus shifting, 811  
 mlib\_MatrixMulSShift\_S16C\_Sat — matrix multiplication by scalar plus shifting, in place, 809  
 mlib\_MatrixMulSShift\_S32\_Mod — matrix multiplication by scalar plus shifting, in place, 809  
 mlib\_MatrixMulSShift\_S32\_S32\_Mod — matrix multiplication by scalar plus shifting, 811  
 mlib\_MatrixMulSShift\_S32\_S32\_Sat — matrix multiplication by scalar plus shifting, 811  
 mlib\_MatrixMulSShift\_S32\_Sat — matrix multiplication by scalar plus shifting, in place, 809  
 mlib\_MatrixMulSShift\_S32C\_Mod — matrix multiplication by scalar plus shifting, in place, 809  
 mlib\_MatrixMulSShift\_S32C\_S32C\_Mod — matrix multiplication by scalar plus shifting, 811  
 mlib\_MatrixMulSShift\_S32C\_S32C\_Sat — matrix multiplication by scalar plus shifting, 811  
 mlib\_MatrixMulSShift\_S32C\_Sat — matrix multiplication by scalar plus shifting, in place, 809  
 mlib\_MatrixMulSShift\_S8\_Mod — matrix multiplication by scalar plus shifting, in place, 809  
 mlib\_MatrixMulSShift\_S8\_S8\_Mod — matrix multiplication by scalar plus shifting, 811  
 mlib\_MatrixMulSShift\_S8\_S8\_Sat — matrix multiplication by scalar plus shifting, 811  
 mlib\_MatrixMulSShift\_S8\_Sat — matrix multiplication by scalar plus shifting, in place, 809  
 mlib\_MatrixMulSShift\_S8C\_Mod — matrix multiplication by scalar plus shifting, in place, 809  
 mlib\_MatrixMulSShift\_S8C\_S8C\_Mod — matrix multiplication by scalar plus shifting, 811  
 mlib\_MatrixMulSShift\_S8C\_S8C\_Sat — matrix multiplication by scalar plus shifting, 811  
 mlib\_MatrixMulSShift\_S8C\_Sat — matrix multiplication by scalar plus shifting, in place, 809  
 mlib\_MatrixMulSShift\_U8\_Mod — matrix multiplication by scalar plus shifting, in place, 809  
 mlib\_MatrixMulSShift\_U8\_Sat — matrix multiplication by scalar plus shifting, in place, 809  
 mlib\_MatrixMulSShift\_U8\_U8\_Mod — matrix multiplication by scalar plus shifting, 811  
 mlib\_MatrixMulSShift\_U8\_U8\_Sat — matrix multiplication by scalar plus shifting, 811  
 mlib\_MatrixMulSShift\_U8C\_Mod — matrix multiplication by scalar plus shifting, in place, 809  
 mlib\_MatrixMulSShift\_U8C\_Sat — matrix multiplication by scalar plus shifting, in place, 809  
 mlib\_MatrixMulSShift\_U8C\_U8C\_Mod — matrix multiplication by scalar plus shifting, 811  
 mlib\_MatrixMulSShift\_U8C\_U8C\_Sat — matrix multiplication by scalar plus shifting, 811  
 mlib\_MatrixScale\_S16\_Mod — matrix linear scaling, in place, 823  
 mlib\_MatrixScale\_S16\_S16\_Mod — matrix linear scaling, 825  
 mlib\_MatrixScale\_S16\_S16\_Sat — matrix linear scaling, 825

mlib\_MatrixScale\_S16\_S8\_Mod — matrix linear scaling, 825  
 mlib\_MatrixScale\_S16\_S8\_Sat — matrix linear scaling, 825  
 mlib\_MatrixScale\_S16\_Sat — matrix linear scaling, in place, 823  
 mlib\_MatrixScale\_S16\_U8\_Mod — matrix linear scaling, 825  
 mlib\_MatrixScale\_S16\_U8\_Sat — matrix linear scaling, 825  
 mlib\_MatrixScale\_S16C\_Mod — matrix linear scaling, in place, 823  
 mlib\_MatrixScale\_S16C\_S16C\_Mod — matrix linear scaling, 825  
 mlib\_MatrixScale\_S16C\_S16C\_Sat — matrix linear scaling, 825  
 mlib\_MatrixScale\_S16C\_S8C\_Mod — matrix linear scaling, 825  
 mlib\_MatrixScale\_S16C\_S8C\_Sat — matrix linear scaling, 825  
 mlib\_MatrixScale\_S16C\_Sat — matrix linear scaling, in place, 823  
 mlib\_MatrixScale\_S16C\_U8C\_Mod — matrix linear scaling, 825  
 mlib\_MatrixScale\_S16C\_U8C\_Sat — matrix linear scaling, 825  
 mlib\_MatrixScale\_S32\_Mod — matrix linear scaling, in place, 823  
 mlib\_MatrixScale\_S32\_S16\_Mod — matrix linear scaling, 825  
 mlib\_MatrixScale\_S32\_S16\_Sat — matrix linear scaling, 825  
 mlib\_MatrixScale\_S32\_S32\_Mod — matrix linear scaling, 825  
 mlib\_MatrixScale\_S32\_S32\_Sat — matrix linear scaling, 825  
 mlib\_MatrixScale\_S32\_Sat — matrix linear scaling, in place, 823  
 mlib\_MatrixScale\_S32C\_Mod — matrix linear scaling, in place, 823  
 mlib\_MatrixScale\_S32C\_S16C\_Mod — matrix linear scaling, 825  
 mlib\_MatrixScale\_S32C\_S16C\_Sat — matrix linear scaling, 825  
 mlib\_MatrixScale\_S32C\_S32C\_Mod — matrix linear scaling, 825  
 mlib\_MatrixScale\_S32C\_S32C\_Sat — matrix linear scaling, 825  
 mlib\_MatrixScale\_S32C\_Sat — matrix linear scaling, in place, 823  
 mlib\_MatrixScale\_S8\_Mod — matrix linear scaling, in place, 823  
 mlib\_MatrixScale\_S8\_S8\_Mod — matrix linear scaling, 825  
 mlib\_MatrixScale\_S8\_S8\_Sat — matrix linear scaling, 825  
 mlib\_MatrixScale\_S8\_Sat — matrix linear scaling, in place, 823  
 mlib\_MatrixScale\_S8C\_Mod — matrix linear scaling, in place, 823  
 mlib\_MatrixScale\_S8C\_S8C\_Mod — matrix linear scaling, 825  
 mlib\_MatrixScale\_S8C\_S8C\_Sat — matrix linear scaling, 825  
 mlib\_MatrixScale\_S8C\_Sat — matrix linear scaling, in place, 823  
 mlib\_MatrixScale\_U8\_Mod — matrix linear scaling, in place, 823  
 mlib\_MatrixScale\_U8\_Sat — matrix linear scaling, in place, 823  
 mlib\_MatrixScale\_U8\_U8\_Mod — matrix linear scaling, 825  
 mlib\_MatrixScale\_U8\_U8\_Sat — matrix linear scaling, 825  
 mlib\_MatrixScale\_U8C\_Mod — matrix linear scaling, in place, 823  
 mlib\_MatrixScale\_U8C\_Sat — matrix linear scaling, in place, 823  
 mlib\_MatrixScale\_U8C\_U8C\_Mod — matrix linear scaling, 825  
 mlib\_MatrixScale\_U8C\_U8C\_Sat — matrix linear scaling, 825  
 mlib\_MatrixSub\_S16\_Mod — matrix subtraction, in place, 834  
 mlib\_MatrixSub\_S16\_S16\_Mod — matrix subtraction, 836  
 mlib\_MatrixSub\_S16\_S16\_Sat — matrix subtraction, 836  
 mlib\_MatrixSub\_S16\_S8\_Mod — matrix subtraction, 836  
 mlib\_MatrixSub\_S16\_S8\_Sat — matrix subtraction, 836  
 mlib\_MatrixSub\_S16\_Sat — matrix subtraction, in place, 834  
 mlib\_MatrixSub\_S16\_U8\_Mod — matrix subtraction, 836

mlib\_MatrixSub\_S16\_U8\_Sat — matrix subtraction, 836  
 mlib\_MatrixSub\_S16C\_Mod — matrix subtraction, in place, 834  
 mlib\_MatrixSub\_S16C\_S16C\_Mod — matrix subtraction, 836  
 mlib\_MatrixSub\_S16C\_S16C\_Sat — matrix subtraction, 836  
 mlib\_MatrixSub\_S16C\_S8C\_Mod — matrix subtraction, 836  
 mlib\_MatrixSub\_S16C\_S8C\_Sat — matrix subtraction, 836  
 mlib\_MatrixSub\_S16C\_Sat — matrix subtraction, in place, 834  
 mlib\_MatrixSub\_S16C\_U8C\_Mod — matrix subtraction, 836  
 mlib\_MatrixSub\_S16C\_U8C\_Sat — matrix subtraction, 836  
 mlib\_MatrixSub\_S32\_Mod — matrix subtraction, in place, 834  
 mlib\_MatrixSub\_S32\_S16\_Mod — matrix subtraction, 836  
 mlib\_MatrixSub\_S32\_S16\_Sat — matrix subtraction, 836  
 mlib\_MatrixSub\_S32\_S32\_Mod — matrix subtraction, 836  
 mlib\_MatrixSub\_S32\_S32\_Sat — matrix subtraction, 836  
 mlib\_MatrixSub\_S32\_Sat — matrix subtraction, in place, 834  
 mlib\_MatrixSub\_S32C\_Mod — matrix subtraction, in place, 834  
 mlib\_MatrixSub\_S32C\_S16C\_Mod — matrix subtraction, 836  
 mlib\_MatrixSub\_S32C\_S16C\_Sat — matrix subtraction, 836  
 mlib\_MatrixSub\_S32C\_S32C\_Mod — matrix subtraction, 836  
 mlib\_MatrixSub\_S32C\_S32C\_Sat — matrix subtraction, 836  
 mlib\_MatrixSub\_S32C\_Sat — matrix subtraction, in place, 834  
 mlib\_MatrixSub\_S8\_Mod — matrix subtraction, in place, 834  
 mlib\_MatrixSub\_S8\_S8\_Mod — matrix subtraction, 836  
 mlib\_MatrixSub\_S8\_S8\_Sat — matrix subtraction, 836  
 mlib\_MatrixSub\_S8\_Sat — matrix subtraction, in place, 834  
 mlib\_MatrixSub\_S8C\_Mod — matrix subtraction, in place, 834  
 mlib\_MatrixSub\_S8C\_S8C\_Mod — matrix subtraction, 836  
 mlib\_MatrixSub\_S8C\_S8C\_Sat — matrix subtraction, 836  
 mlib\_MatrixSub\_S8C\_Sat — matrix subtraction, in place, 834  
 mlib\_MatrixSub\_U8\_Mod — matrix subtraction, in place, 834  
 mlib\_MatrixSub\_U8\_Sat — matrix subtraction, in place, 834  
 mlib\_MatrixSub\_U8\_U8\_Mod — matrix subtraction, 836  
 mlib\_MatrixSub\_U8\_U8\_Sat — matrix subtraction, 836  
 mlib\_MatrixSub\_U8C\_Mod — matrix subtraction, in place, 834  
 mlib\_MatrixSub\_U8C\_Sat — matrix subtraction, in place, 834  
 mlib\_MatrixSub\_U8C\_U8C\_Mod — matrix subtraction, 836  
 mlib\_MatrixSub\_U8C\_U8C\_Sat — matrix subtraction, 836  
 mlib\_MatrixSubS\_S16\_Mod — matrix subtraction from scalar, in place, 829  
 mlib\_MatrixSubS\_S16\_S16\_Mod — matrix subtraction from scalar, 831  
 mlib\_MatrixSubS\_S16\_S16\_Sat — matrix subtraction from scalar, 831  
 mlib\_MatrixSubS\_S16\_S8\_Mod — matrix subtraction from scalar, 831  
 mlib\_MatrixSubS\_S16\_S8\_Sat — matrix subtraction from scalar, 831  
 mlib\_MatrixSubS\_S16\_Sat — matrix subtraction from scalar, in place, 829  
 mlib\_MatrixSubS\_S16\_U8\_Mod — matrix subtraction from scalar, 831  
 mlib\_MatrixSubS\_S16\_U8\_Sat — matrix subtraction from scalar, 831  
 mlib\_MatrixSubS\_S16C\_Mod — matrix subtraction from scalar, in place, 829  
 mlib\_MatrixSubS\_S16C\_S16C\_Mod — matrix subtraction from scalar, 831  
 mlib\_MatrixSubS\_S16C\_S16C\_Sat — matrix subtraction from scalar, 831

mlib\_MatrixSubS\_S16C\_S8C\_Mod — matrix subtraction from scalar, 831  
 mlib\_MatrixSubS\_S16C\_S8C\_Sat — matrix subtraction from scalar, 831  
 mlib\_MatrixSubS\_S16C\_Sat — matrix subtraction from scalar, in place, 829  
 mlib\_MatrixSubS\_S16C\_U8C\_Mod — matrix subtraction from scalar, 831  
 mlib\_MatrixSubS\_S16C\_U8C\_Sat — matrix subtraction from scalar, 831  
 mlib\_MatrixSubS\_S32\_Mod — matrix subtraction from scalar, in place, 829  
 mlib\_MatrixSubS\_S32\_S16\_Mod — matrix subtraction from scalar, 831  
 mlib\_MatrixSubS\_S32\_S16\_Sat — matrix subtraction from scalar, 831  
 mlib\_MatrixSubS\_S32\_S32\_Mod — matrix subtraction from scalar, 831  
 mlib\_MatrixSubS\_S32\_S32\_Sat — matrix subtraction from scalar, 831  
 mlib\_MatrixSubS\_S32\_Sat — matrix subtraction from scalar, in place, 829  
 mlib\_MatrixSubS\_S32C\_Mod — matrix subtraction from scalar, in place, 829  
 mlib\_MatrixSubS\_S32C\_S16C\_Mod — matrix subtraction from scalar, 831  
 mlib\_MatrixSubS\_S32C\_S16C\_Sat — matrix subtraction from scalar, 831  
 mlib\_MatrixSubS\_S32C\_S32C\_Mod — matrix subtraction from scalar, 831  
 mlib\_MatrixSubS\_S32C\_S32C\_Sat — matrix subtraction from scalar, 831  
 mlib\_MatrixSubS\_S32C\_Sat — matrix subtraction from scalar, in place, 829  
 mlib\_MatrixSubS\_S8\_Mod — matrix subtraction from scalar, in place, 829  
 mlib\_MatrixSubS\_S8\_S8\_Mod — matrix subtraction from scalar, 831  
 mlib\_MatrixSubS\_S8\_S8\_Sat — matrix subtraction from scalar, 831  
 mlib\_MatrixSubS\_S8\_Sat — matrix subtraction from scalar, in place, 829  
 mlib\_MatrixSubS\_S8C\_Mod — matrix subtraction from scalar, in place, 829  
 mlib\_MatrixSubS\_S8C\_S8C\_Mod — matrix subtraction from scalar, 831  
 mlib\_MatrixSubS\_S8C\_S8C\_Sat — matrix subtraction from scalar, 831  
 mlib\_MatrixSubS\_S8C\_Sat — matrix subtraction from scalar, in place, 829  
 mlib\_MatrixSubS\_U8\_Mod — matrix subtraction from scalar, in place, 829  
 mlib\_MatrixSubS\_U8\_Sat — matrix subtraction from scalar, in place, 829  
 mlib\_MatrixSubS\_U8\_U8\_Mod — matrix subtraction from scalar, 831  
 mlib\_MatrixSubS\_U8\_U8\_Sat — matrix subtraction from scalar, 831  
 mlib\_MatrixSubS\_U8C\_Mod — matrix subtraction from scalar, in place, 829  
 mlib\_MatrixSubS\_U8C\_Sat — matrix subtraction from scalar, in place, 829  
 mlib\_MatrixSubS\_U8C\_U8C\_Mod — matrix subtraction from scalar, 831  
 mlib\_MatrixSubS\_U8C\_U8C\_Sat — matrix subtraction from scalar, 831  
 mlib\_MatrixTranspose\_S16 — matrix transpose, in place, 839  
 mlib\_MatrixTranspose\_S16\_S16 — matrix transpose, 841  
 mlib\_MatrixTranspose\_S16C — matrix transpose, in place, 839  
 mlib\_MatrixTranspose\_S16C\_S16C — matrix transpose, 841  
 mlib\_MatrixTranspose\_S32 — matrix transpose, in place, 839  
 mlib\_MatrixTranspose\_S32\_S32 — matrix transpose, 841  
 mlib\_MatrixTranspose\_S32C — matrix transpose, in place, 839  
 mlib\_MatrixTranspose\_S32C\_S32C — matrix transpose, 841  
 mlib\_MatrixTranspose\_S8 — matrix transpose, in place, 839  
 mlib\_MatrixTranspose\_S8\_S8 — matrix transpose, 841  
 mlib\_MatrixTranspose\_S8C — matrix transpose, in place, 839  
 mlib\_MatrixTranspose\_S8C\_S8C — matrix transpose, 841  
 mlib\_MatrixTranspose\_U8 — matrix transpose, in place, 839  
 mlib\_MatrixTranspose\_U8\_U8 — matrix transpose, 841  
 mlib\_MatrixTranspose\_U8C — matrix transpose, in place, 839

mlib\_MatrixTranspose\_U8C\_U8C — matrix transpose, 841  
 mlib\_MatrixUnit\_S16 — Unit matrix generation, 843  
 mlib\_MatrixUnit\_S16C — Unit matrix generation, 843  
 mlib\_MatrixUnit\_S32 — Unit matrix generation, 843  
 mlib\_MatrixUnit\_S32C — Unit matrix generation, 843  
 mlib\_MatrixUnit\_S8 — Unit matrix generation, 843  
 mlib\_MatrixUnit\_S8C — Unit matrix generation, 843  
 mlib\_MatrixUnit\_U8 — Unit matrix generation, 843  
 mlib\_MatrixUnit\_U8C — Unit matrix generation, 843  
 mlib\_memcpy — copy a block of bytes, 845  
 mlib\_memmove — copy a block of bytes, 846  
 mlib\_memset — set a block of bytes, 847  
 mlib\_realloc — reallocate a block of bytes, 848  
 mlib\_SignalADPCM2Bits2Linear — adaptive differential pulse code modulation (ADPCM), 849  
 mlib\_SignalADPCM3Bits2Linear — adaptive differential pulse code modulation (ADPCM), 850  
 mlib\_SignalADPCM4Bits2Linear — adaptive differential pulse code modulation (ADPCM), 851  
 mlib\_SignalADPCM5Bits2Linear — adaptive differential pulse code modulation (ADPCM), 852  
 mlib\_SignalADPCMFree — adaptive differential pulse code modulation (ADPCM), 853  
 mlib\_SignalADPCMInit — adaptive differential pulse code modulation (ADPCM), 854  
 mlib\_SignalALaw2Linear — ITU G.711 m-law and A-law compression and decompression, 855  
 mlib\_SignalALaw2uLaw — ITU G.711 m-law and A-law compression and decompression, 856  
 mlib\_SignalAutoCorrel\_F32 — signal auto-correlation, 857  
 mlib\_SignalAutoCorrel\_F32S — signal auto-correlation, 857  
 mlib\_SignalAutoCorrel\_S16 — signal auto-correlation, 857  
 mlib\_SignalAutoCorrel\_S16S — signal auto-correlation, 857  
 mlib\_SignalCepstral\_F32 — perform cepstral analysis, 859  
 mlib\_SignalCepstral\_S16 — perform cepstral analysis, 863  
 mlib\_SignalCepstral\_S16\_Adp — perform cepstral analysis, 865  
 mlib\_SignalCepstralFree\_F32 — clean up for cepstral analysis, 861  
 mlib\_SignalCepstralFree\_S16 — clean up for cepstral analysis, 861  
 mlib\_SignalCepstralInit\_F32 — initialization for cepstral analysis, 862  
 mlib\_SignalCepstralInit\_S16 — initialization for cepstral analysis, 862  
 mlib\_SignalConv\_F32\_F32 — signal convolution, 873  
 mlib\_SignalConv\_F32S\_F32S — signal convolution, 873  
 mlib\_SignalConv\_S16\_S16\_Sat — signal convolution, 873  
 mlib\_SignalConv\_S16S\_S16S\_Sat — signal convolution, 873  
 mlib\_SignalConvertShift\_F32\_S16 — data type convert with shifting, 867  
 mlib\_SignalConvertShift\_F32\_S32 — data type convert with shifting, 867  
 mlib\_SignalConvertShift\_F32\_S8 — data type convert with shifting, 867  
 mlib\_SignalConvertShift\_F32\_U8 — data type convert with shifting, 867  
 mlib\_SignalConvertShift\_F32S\_S16S — data type convert with shifting, 867  
 mlib\_SignalConvertShift\_F32S\_S32S — data type convert with shifting, 867  
 mlib\_SignalConvertShift\_F32S\_S8S — data type convert with shifting, 867  
 mlib\_SignalConvertShift\_F32S\_U8S — data type convert with shifting, 867  
 mlib\_SignalConvertShift\_S16\_F32\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_S16\_S32\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_S16\_S8\_Sat — data type convert with shifting, 869

mlib\_SignalConvertShift\_S16\_U8\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_S16S\_F32S\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_S16S\_S32S\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_S16S\_S8S\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_S16S\_U8S\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_S32\_F32\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_S32\_S16\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_S32\_S8\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_S32\_U8\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_S32S\_F32S\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_S32S\_S16S\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_S32S\_S8S\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_S32S\_U8S\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_S8\_F32\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_S8\_S16\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_S8\_S32\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_S8\_U8\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_S8S\_F32S\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_S8S\_S16S\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_S8S\_S32S\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_S8S\_U8S\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_U8\_F32\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_U8\_S16\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_U8\_S8\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_U8S\_F32S\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_U8S\_S16S\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_U8S\_S32S\_Sat — data type convert with shifting, 869  
 mlib\_SignalConvertShift\_U8S\_S8S\_Sat — data type convert with shifting, 869  
 mlib\_SignalCrossCorrel\_F32 — signal cross correlation, 875  
 mlib\_SignalCrossCorrel\_F32S — signal cross correlation, 875  
 mlib\_SignalCrossCorrel\_S16 — signal cross correlation, 875  
 mlib\_SignalCrossCorrel\_S16S — signal cross correlation, 875  
 mlib\_SignalDownSample\_F32\_F32 — signal downsampling, 877  
 mlib\_SignalDownSample\_F32S\_F32S — signal downsampling, 877  
 mlib\_SignalDownSample\_S16\_S16 — signal downsampling, 877  
 mlib\_SignalDownSample\_S16S\_S16S — signal downsampling, 877  
 mlib\_SignalDTWKScalar\_F32 — perform dynamic time warping for K-best paths on scalar data, 879  
 mlib\_SignalDTWKScalar\_S16 — perform dynamic time warping for K-best paths on scalar data, 890  
 mlib\_SignalDTWKScalarFree\_F32 — clean up for K-best paths of scalar data, 883  
 mlib\_SignalDTWKScalarFree\_S16 — clean up for K-best paths of scalar data, 883  
 mlib\_SignalDTWKScalarInit\_F32 — initialization for K-best paths of scalar data, 884  
 mlib\_SignalDTWKScalarInit\_S16 — initialization for K-best paths of scalar data, 885  
 mlib\_SignalDTWKScalarPath\_F32 — return K-best path on scalar data, 886  
 mlib\_SignalDTWKScalarPath\_S16 — return K-best path on scalar data, 886



mlib\_SignalDTWKVector\_F32 — perform dynamic time warping for K-best paths on vector data, 894  
 mlib\_SignalDTWKVector\_S16 — perform dynamic time warping for K-best paths on vector data, 907  
 mlib\_SignalDTWKVectorFree\_F32 — clean up for K-best paths of vector data, 898  
 mlib\_SignalDTWKVectorFree\_S16 — clean up for K-best paths of vector data, 898  
 mlib\_SignalDTWKVectorInit\_F32 — initialization for K-best paths of vector data, 899  
 mlib\_SignalDTWKVectorInit\_S16 — initialization for K-best paths of vector data, 901  
 mlib\_SignalDTWKVectorPath\_F32 — return K-best path on vector data, 903  
 mlib\_SignalDTWKVectorPath\_S16 — return K-best path on vector data, 903  
 mlib\_SignalDTWScalar\_F32 — perform dynamic time warping on scalar data, 911  
 mlib\_SignalDTWScalar\_S16 — perform dynamic time warping on scalar data, 926  
 mlib\_SignalDTWScalarFree\_F32 — clean up for scalar data, 915  
 mlib\_SignalDTWScalarFree\_S16 — clean up for scalar data, 915  
 mlib\_SignalDTWScalarInit\_F32 — initialization for scalar data, 916  
 mlib\_SignalDTWScalarInit\_S16 — initialization for scalar data, 917  
 mlib\_SignalDTWScalarPath\_F32 — perform dynamic time warping on scalar data, 918  
 mlib\_SignalDTWScalarPath\_S16 — perform dynamic time warping on scalar data, 922  
 mlib\_SignalDTWVector\_F32 — perform dynamic time warping on vector data, 930  
 mlib\_SignalDTWVector\_S16 — perform dynamic time warping on vector data, 946  
 mlib\_SignalDTWVectorFree\_F32 — clean up for vector data, 934  
 mlib\_SignalDTWVectorFree\_S16 — clean up for vector data, 934  
 mlib\_SignalDTWVectorInit\_F32 — initialization for vector data, 935  
 mlib\_SignalDTWVectorInit\_S16 — initialization for vector data, 936  
 mlib\_SignalDTWVectorPath\_F32 — perform dynamic time warping on vector data, 938  
 mlib\_SignalDTWVectorPath\_S16 — perform dynamic time warping on vector data, 942  
 mlib\_SignalEmphasize\_F32\_F32 — signal pre-emphasizing, 952  
 mlib\_SignalEmphasize\_F32S\_F32S — signal pre-emphasizing, 952  
 mlib\_SignalEmphasize\_S16\_S16\_Sat — signal pre-emphasizing, 952  
 mlib\_SignalEmphasize\_S16S\_S16S\_Sat — signal pre-emphasizing, 952  
 mlib\_SignalEmphasizeFree\_F32\_F32 — clean up for signal pre-emphasizing, 950  
 mlib\_SignalEmphasizeFree\_F32S\_F32S — clean up for signal pre-emphasizing, 950  
 mlib\_SignalEmphasizeFree\_S16\_S16 — clean up for signal pre-emphasizing, 950  
 mlib\_SignalEmphasizeFree\_S16S\_S16S — clean up for signal pre-emphasizing, 950  
 mlib\_SignalEmphasizeInit\_F32\_F32 — initialization for signal pre-emphasizing, 951  
 mlib\_SignalEmphasizeInit\_F32S\_F32S — initialization for signal pre-emphasizing, 951  
 mlib\_SignalEmphasizeInit\_S16\_S16 — initialization for signal pre-emphasizing, 951  
 mlib\_SignalEmphasizeInit\_S16S\_S16S — initialization for signal pre-emphasizing, 951  
 mlib\_SignalFFT\_1 — signal Fast Fourier Transform (FFT), 954  
 mlib\_SignalFFT\_1\_D64 — signal Fast Fourier Transform (FFT), 954  
 mlib\_SignalFFT\_1\_D64\_D64 — signal Fast Fourier Transform (FFT), 954  
 mlib\_SignalFFT\_1\_D64C — signal Fast Fourier Transform (FFT), 954  
 mlib\_SignalFFT\_1\_D64C\_D64 — signal Fast Fourier Transform (FFT), 954  
 mlib\_SignalFFT\_1\_D64C\_D64C — signal Fast Fourier Transform (FFT), 954  
 mlib\_SignalFFT\_1\_F32 — signal Fast Fourier Transform (FFT), 954  
 mlib\_SignalFFT\_1\_F32\_F32 — signal Fast Fourier Transform (FFT), 954  
 mlib\_SignalFFT\_1\_F32C — signal Fast Fourier Transform (FFT), 954  
 mlib\_SignalFFT\_1\_F32C\_F32 — signal Fast Fourier Transform (FFT), 954

mlib\_SignalFFT\_1\_F32C\_F32C — signal Fast Fourier Transform (FFT), 954  
 mlib\_SignalFFT\_1\_S16\_Mod — signal Fast Fourier Transform (FFT), 954  
 mlib\_SignalFFT\_1\_S16\_S16\_Mod — signal Fast Fourier Transform (FFT), 954  
 mlib\_SignalFFT\_1\_S16C\_Mod — signal Fast Fourier Transform (FFT), 954  
 mlib\_SignalFFT\_1\_S16C\_S16\_Mod — signal Fast Fourier Transform (FFT), 954  
 mlib\_SignalFFT\_1\_S16C\_S16C\_Mod — signal Fast Fourier Transform (FFT), 954  
 mlib\_SignalFFT\_2 — signal Fast Fourier Transform (FFT), 957  
 mlib\_SignalFFT\_2\_D64 — signal Fast Fourier Transform (FFT), 957  
 mlib\_SignalFFT\_2\_D64\_D64 — signal Fast Fourier Transform (FFT), 957  
 mlib\_SignalFFT\_2\_D64C — signal Fast Fourier Transform (FFT), 957  
 mlib\_SignalFFT\_2\_D64C\_D64 — signal Fast Fourier Transform (FFT), 957  
 mlib\_SignalFFT\_2\_D64C\_D64C — signal Fast Fourier Transform (FFT), 957  
 mlib\_SignalFFT\_2\_F32 — signal Fast Fourier Transform (FFT), 957  
 mlib\_SignalFFT\_2\_F32\_F32 — signal Fast Fourier Transform (FFT), 957  
 mlib\_SignalFFT\_2\_F32C — signal Fast Fourier Transform (FFT), 957  
 mlib\_SignalFFT\_2\_F32C\_F32 — signal Fast Fourier Transform (FFT), 957  
 mlib\_SignalFFT\_2\_F32C\_F32C — signal Fast Fourier Transform (FFT), 957  
 mlib\_SignalFFT\_2\_S16 — signal Fast Fourier Transform (FFT), 957  
 mlib\_SignalFFT\_2\_S16\_S16 — signal Fast Fourier Transform (FFT), 957  
 mlib\_SignalFFT\_2\_S16C — signal Fast Fourier Transform (FFT), 957  
 mlib\_SignalFFT\_2\_S16C\_S16 — signal Fast Fourier Transform (FFT), 957  
 mlib\_SignalFFT\_2\_S16C\_S16C — signal Fast Fourier Transform (FFT), 957  
 mlib\_SignalFFT\_3 — signal Fast Fourier Transform (FFT), 960  
 mlib\_SignalFFT\_3\_D64 — signal Fast Fourier Transform (FFT), 960  
 mlib\_SignalFFT\_3\_D64\_D64 — signal Fast Fourier Transform (FFT), 960  
 mlib\_SignalFFT\_3\_D64C — signal Fast Fourier Transform (FFT), 960  
 mlib\_SignalFFT\_3\_D64C\_D64 — signal Fast Fourier Transform (FFT), 960  
 mlib\_SignalFFT\_3\_D64C\_D64C — signal Fast Fourier Transform (FFT), 960  
 mlib\_SignalFFT\_3\_F32 — signal Fast Fourier Transform (FFT), 960  
 mlib\_SignalFFT\_3\_F32\_F32 — signal Fast Fourier Transform (FFT), 960  
 mlib\_SignalFFT\_3\_F32C — signal Fast Fourier Transform (FFT), 960  
 mlib\_SignalFFT\_3\_F32C\_F32 — signal Fast Fourier Transform (FFT), 960  
 mlib\_SignalFFT\_3\_F32C\_F32C — signal Fast Fourier Transform (FFT), 960  
 mlib\_SignalFFT\_3\_S16\_Mod — signal Fast Fourier Transform (FFT), 960  
 mlib\_SignalFFT\_3\_S16\_S16\_Mod — signal Fast Fourier Transform (FFT), 960  
 mlib\_SignalFFT\_3\_S16C\_Mod — signal Fast Fourier Transform (FFT), 960  
 mlib\_SignalFFT\_3\_S16C\_S16\_Mod — signal Fast Fourier Transform (FFT), 960  
 mlib\_SignalFFT\_3\_S16C\_S16C\_Mod — signal Fast Fourier Transform (FFT), 960  
 mlib\_SignalFFT\_4 — signal Fast Fourier Transform (FFT), 963  
 mlib\_SignalFFT\_4\_S16 — signal Fast Fourier Transform (FFT), 963  
 mlib\_SignalFFT\_4\_S16\_S16 — signal Fast Fourier Transform (FFT), 963  
 mlib\_SignalFFT\_4\_S16C — signal Fast Fourier Transform (FFT), 963  
 mlib\_SignalFFT\_4\_S16C\_S16 — signal Fast Fourier Transform (FFT), 963  
 mlib\_SignalFFT\_4\_S16C\_S16C — signal Fast Fourier Transform (FFT), 963  
 mlib\_SignalFFTW\_1 — signal Fast Fourier Transform with windowing (FFTW), 965  
 mlib\_SignalFFTW\_1\_F32 — signal Fast Fourier Transform with windowing (FFTW), 965  
 mlib\_SignalFFTW\_1\_F32\_F32 — signal Fast Fourier Transform with windowing (FFTW), 965



mlib\_SignalFFTW\_4\_S16C\_S16 — signal Fast Fourier Transform with windowing (FFTW), 974  
 mlib\_SignalFFTW\_4\_S16C\_S16C — signal Fast Fourier Transform with windowing (FFTW), 974  
 mlib\_SignalFIR\_F32\_F32 — Finite Impulse Response (FIR) filtering, 977  
 mlib\_SignalFIR\_F32S\_F32S — Finite Impulse Response (FIR) filtering, 978  
 mlib\_SignalFIR\_S16\_S16\_Sat — Finite Impulse Response (FIR) filtering, 985  
 mlib\_SignalFIR\_S16S\_S16S\_Sat — Finite Impulse Response (FIR) filtering, 985  
 mlib\_SignalFIRFree\_F32\_F32 — Finite Impulse Response (FIR) filtering, 979  
 mlib\_SignalFIRFree\_F32S\_F32S — Finite Impulse Response (FIR) filtering, 980  
 mlib\_SignalFIRFree\_S16\_S16 — Finite Impulse Response (FIR) filtering, 981  
 mlib\_SignalFIRFree\_S16S\_S16S — Finite Impulse Response (FIR) filtering, 981  
 mlib\_SignalFIRInit\_F32\_F32 — Finite Impulse Response (FIR) filtering, 982  
 mlib\_SignalFIRInit\_F32S\_F32S — Finite Impulse Response (FIR) filtering, 983  
 mlib\_SignalFIRInit\_S16\_S16 — Finite Impulse Response (FIR) filtering, 984  
 mlib\_SignalFIRInit\_S16S\_S16S — Finite Impulse Response (FIR) filtering, 984  
 mlib\_SignalGaussNoise\_F32 — Gaussian noise generation, 986  
 mlib\_SignalGaussNoise\_S16 — Gaussian noise generation, 991  
 mlib\_SignalGaussNoiseFree\_F32 — Gaussian noise generation, 987  
 mlib\_SignalGaussNoiseFree\_S16 — Gaussian noise generation, 988  
 mlib\_SignalGaussNoiseInit\_F32 — Gaussian noise generation, 989  
 mlib\_SignalGaussNoiseInit\_S16 — Gaussian noise generation, 990  
 mlib\_SignalGenBartlett\_F32 — Bartlett window generation, 992  
 mlib\_SignalGenBartlett\_S16 — Bartlett window generation, 993  
 mlib\_SignalGenBlackman\_F32 — Blackman window generation, 994  
 mlib\_SignalGenBlackman\_S16 — Blackman window generation, 995  
 mlib\_SignalGenHamming\_F32 — Hamming window generation, 996  
 mlib\_SignalGenHamming\_S16 — Hamming window generation, 997  
 mlib\_SignalGenHanning\_F32 — Hanning window generation, 998  
 mlib\_SignalGenHanning\_S16 — Hanning window generation, 999  
 mlib\_SignalGenKaiser\_F32 — Kaiser window generation, 1000  
 mlib\_SignalGenKaiser\_S16 — Kaiser window generation, 1001  
 mlib\_SignalIFFT\_1 — signal Inverse Fast Fourier Transform (IFFT), 1002  
 mlib\_SignalIFFT\_1\_D64 — signal Inverse Fast Fourier Transform (IFFT), 1002  
 mlib\_SignalIFFT\_1\_D64\_D64 — signal Inverse Fast Fourier Transform (IFFT), 1002  
 mlib\_SignalIFFT\_1\_D64\_D64C — signal Inverse Fast Fourier Transform (IFFT), 1002  
 mlib\_SignalIFFT\_1\_D64C — signal Inverse Fast Fourier Transform (IFFT), 1002  
 mlib\_SignalIFFT\_1\_D64C\_D64C — signal Inverse Fast Fourier Transform (IFFT), 1002  
 mlib\_SignalIFFT\_1\_F32 — signal Inverse Fast Fourier Transform (IFFT), 1002  
 mlib\_SignalIFFT\_1\_F32\_F32 — signal Inverse Fast Fourier Transform (IFFT), 1002  
 mlib\_SignalIFFT\_1\_F32\_F32C — signal Inverse Fast Fourier Transform (IFFT), 1002  
 mlib\_SignalIFFT\_1\_F32C — signal Inverse Fast Fourier Transform (IFFT), 1002  
 mlib\_SignalIFFT\_1\_F32C\_F32C — signal Inverse Fast Fourier Transform (IFFT), 1002  
 mlib\_SignalIFFT\_1\_S16 — signal Inverse Fast Fourier Transform (IFFT), 1002  
 mlib\_SignalIFFT\_1\_S16\_S16 — signal Inverse Fast Fourier Transform (IFFT), 1002  
 mlib\_SignalIFFT\_1\_S16\_S16C — signal Inverse Fast Fourier Transform (IFFT), 1002  
 mlib\_SignalIFFT\_1\_S16C — signal Inverse Fast Fourier Transform (IFFT), 1002  
 mlib\_SignalIFFT\_1\_S16C\_S16C — signal Inverse Fast Fourier Transform (IFFT), 1002  
 mlib\_SignalIFFT\_2 — signal Inverse Fast Fourier Transform (IFFT), 1005



mlib\_SignalIFFTW\_1\_S16\_S16 — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1013  
 mlib\_SignalIFFTW\_1\_S16\_S16C — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1013  
 mlib\_SignalIFFTW\_1\_S16C — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1013  
 mlib\_SignalIFFTW\_1\_S16C\_S16C — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1013  
 mlib\_SignalIFFTW\_2 — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1016  
 mlib\_SignalIFFTW\_2\_F32 — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1016  
 mlib\_SignalIFFTW\_2\_F32\_F32 — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1016  
 mlib\_SignalIFFTW\_2\_F32\_F32C — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1016  
 mlib\_SignalIFFTW\_2\_F32C — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1016  
 mlib\_SignalIFFTW\_2\_F32C\_F32C — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1016  
 mlib\_SignalIFFTW\_2\_S16\_Mod — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1016  
 mlib\_SignalIFFTW\_2\_S16\_S16\_Mod — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1016  
 mlib\_SignalIFFTW\_2\_S16\_S16C\_Mod — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1016  
 mlib\_SignalIFFTW\_2\_S16C\_Mod — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1016  
 mlib\_SignalIFFTW\_2\_S16C\_S16C\_Mod — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1016  
 mlib\_SignalIFFTW\_3 — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1019  
 mlib\_SignalIFFTW\_3\_F32 — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1019  
 mlib\_SignalIFFTW\_3\_F32\_F32 — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1019  
 mlib\_SignalIFFTW\_3\_F32\_F32C — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1019  
 mlib\_SignalIFFTW\_3\_F32C — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1019  
 mlib\_SignalIFFTW\_3\_F32C\_F32C — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1019  
 mlib\_SignalIFFTW\_3\_S16\_Mod — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1019  
 mlib\_SignalIFFTW\_3\_S16\_S16\_Mod — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1019  
 mlib\_SignalIFFTW\_3\_S16\_S16C\_Mod — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1019  
 mlib\_SignalIFFTW\_3\_S16C\_Mod — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1019  
 mlib\_SignalIFFTW\_3\_S16C\_S16C\_Mod — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1019  
 mlib\_SignalIFFTW\_4 — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1022  
 mlib\_SignalIFFTW\_4\_S16 — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1022  
 mlib\_SignalIFFTW\_4\_S16\_S16 — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1022  
 mlib\_SignalIFFTW\_4\_S16\_S16C — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1022  
 mlib\_SignalIFFTW\_4\_S16C — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1022  
 mlib\_SignalIFFTW\_4\_S16C\_S16C — signal Inverse Fast Fourier Transform with windowing (IFFTW), 1022

mlib\_SignalIIR\_Biquad\_F32\_F32 — biquad Infinite Impulse Response (IIR) filtering, 1025  
 mlib\_SignalIIR\_Biquad\_F32S\_F32S — biquad Infinite Impulse Response (IIR) filtering, 1025  
 mlib\_SignalIIR\_Biquad\_S16\_S16\_Sat — biquad Infinite Impulse Response (IIR) filtering, 1025  
 mlib\_SignalIIR\_Biquad\_S16S\_S16S\_Sat — biquad Infinite Impulse Response (IIR) filtering, 1025  
 mlib\_SignalIIR\_P4\_F32\_F32 — parallel Infinite Impulse Response (IIR) filtering, 1035  
 mlib\_SignalIIR\_P4\_F32S\_F32S — parallel Infinite Impulse Response (IIR) filtering, 1035  
 mlib\_SignalIIR\_P4\_S16\_S16\_Sat — parallel Infinite Impulse Response (IIR) filtering, 1035  
 mlib\_SignalIIR\_P4\_S16S\_S16S\_Sat — parallel Infinite Impulse Response (IIR) filtering, 1035  
 mlib\_SignalIIRFree\_Biquad\_F32\_F32 — biquad Infinite Impulse Response (IIR) filtering, 1027  
 mlib\_SignalIIRFree\_Biquad\_F32S\_F32S — biquad Infinite Impulse Response (IIR) filtering, 1027  
 mlib\_SignalIIRFree\_Biquad\_S16\_S16 — biquad Infinite Impulse Response (IIR) filtering, 1028  
 mlib\_SignalIIRFree\_Biquad\_S16S\_S16S — biquad Infinite Impulse Response (IIR) filtering, 1028  
 mlib\_SignalIIRFree\_P4\_F32\_F32 — parallel Infinite Impulse Response (IIR) filtering, 1029  
 mlib\_SignalIIRFree\_P4\_F32S\_F32S — parallel Infinite Impulse Response (IIR) filtering, 1029  
 mlib\_SignalIIRFree\_P4\_S16\_S16 — parallel Infinite Impulse Response (IIR) filtering, 1030  
 mlib\_SignalIIRFree\_P4\_S16S\_S16S — parallel Infinite Impulse Response (IIR) filtering, 1030  
 mlib\_SignalIIRInit\_Biquad\_F32\_F32 — biquad Infinite Impulse Response (IIR) filtering, 1031  
 mlib\_SignalIIRInit\_Biquad\_F32S\_F32S — biquad Infinite Impulse Response (IIR) filtering, 1031  
 mlib\_SignalIIRInit\_Biquad\_S16\_S16 — biquad Infinite Impulse Response (IIR) filtering, 1032  
 mlib\_SignalIIRInit\_Biquad\_S16S\_S16S — biquad Infinite Impulse Response (IIR) filtering, 1032  
 mlib\_SignalIIRInit\_P4\_F32\_F32 — parallel Infinite Impulse Response (IIR) filtering, 1033  
 mlib\_SignalIIRInit\_P4\_F32S\_F32S — parallel Infinite Impulse Response (IIR) filtering, 1033  
 mlib\_SignalIIRInit\_P4\_S16\_S16 — parallel Infinite Impulse Response (IIR) filtering, 1034  
 mlib\_SignalIIRInit\_P4\_S16S\_S16S — parallel Infinite Impulse Response (IIR) filtering, 1034  
 mlib\_SignalIMDCT\_D64 — Dolby AC-3 digital audio standard transformation, 1038  
 mlib\_SignalIMDCT\_F32 — Dolby AC-3 digital audio standard transformation, 1039  
 mlib\_SignalIMDCTSplit\_D64 — Dolby AC-3 digital audio standard transformation, 1040  
 mlib\_SignalIMDCTSplit\_F32 — Dolby AC-3 digital audio standard transformation, 1041  
 mlib\_SignalLimit — signal hard limiting, 1042  
 mlib\_SignalLimit\_F32 — signal hard limiting, 1042  
 mlib\_SignalLimit\_F32\_F32 — signal hard limiting, 1042  
 mlib\_SignalLimit\_F32S — signal hard limiting, 1042  
 mlib\_SignalLimit\_F32S\_F32S — signal hard limiting, 1042  
 mlib\_SignalLimit\_S16 — signal hard limiting, 1042  
 mlib\_SignalLimit\_S16\_S16 — signal hard limiting, 1042  
 mlib\_SignalLimit\_S16S — signal hard limiting, 1042

`mllib_SignalLimit_S16S_S16S` — signal hard limiting, 1042  
`mllib_SignalLinear2ADPCM2Bits` — adaptive differential pulse code modulation (ADPCM), 1044  
`mllib_SignalLinear2ADPCM3Bits` — adaptive differential pulse code modulation (ADPCM), 1045  
`mllib_SignalLinear2ADPCM4Bits` — adaptive differential pulse code modulation (ADPCM), 1046  
`mllib_SignalLinear2ADPCM5Bits` — adaptive differential pulse code modulation (ADPCM), 1047  
`mllib_SignalLinear2ALaw` — ITU G.711 m-law and A-law compression and decompression, 1048  
`mllib_SignalLinear2uLaw` — ITU G.711 m-law and A-law compression and decompression, 1049  
`mllib_SignalLMSFilter_F32_F32` — least mean square (LMS) adaptive filtering, 1050  
`mllib_SignalLMSFilter_F32S_F32S` — least mean square (LMS) adaptive filtering, 1050  
`mllib_SignalLMSFilter_S16_S16_Sat` — least mean square (LMS) adaptive filtering, 1055  
`mllib_SignalLMSFilter_S16S_S16S_Sat` — least mean square (LMS) adaptive filtering, 1055  
`mllib_SignalLMSFilterFree_F32_F32` — least mean square (LMS) adaptive filtering, 1051  
`mllib_SignalLMSFilterFree_F32S_F32S` — least mean square (LMS) adaptive filtering, 1051  
`mllib_SignalLMSFilterFree_S16_S16` — least mean square (LMS) adaptive filtering, 1052  
`mllib_SignalLMSFilterFree_S16S_S16S` — least mean square (LMS) adaptive filtering, 1052  
`mllib_SignalLMSFilterInit_F32_F32` — least mean square (LMS) adaptive filtering, 1053  
`mllib_SignalLMSFilterInit_F32S_F32S` — least mean square (LMS) adaptive filtering, 1053  
`mllib_SignalLMSFilterInit_S16_S16` — least mean square (LMS) adaptive filtering, 1054  
`mllib_SignalLMSFilterInit_S16S_S16S` — least mean square (LMS) adaptive filtering, 1054  
`mllib_SignalLPC2Cepstral_F32` — convert linear prediction coefficients to cepstral coefficients, 1056  
`mllib_SignalLPC2Cepstral_S16` — convert linear prediction coefficients to cepstral coefficients, 1058  
`mllib_SignalLPC2Cepstral_S16_Adap` — convert linear prediction coefficients to cepstral coefficients, 1060  
`mllib_SignalLPC2LSP_F32` — convert linear prediction coefficients to line spectral pair coefficients, 1062  
`mllib_SignalLPC2LSP_S16` — convert linear prediction coefficients to line spectral pair coefficients, 1064  
`mllib_SignalLPCAutoCorrel_F32` — perform linear predictive coding with autocorrelation method, 1066  
`mllib_SignalLPCAutoCorrel_S16` — perform linear predictive coding with autocorrelation method, 1078  
`mllib_SignalLPCAutoCorrel_S16_Adap` — perform linear predictive coding with autocorrelation method, 1078  
`mllib_SignalLPCAutoCorrelFree_F32` — clean up for autocorrelation method, 1068  
`mllib_SignalLPCAutoCorrelFree_S16` — clean up for autocorrelation method, 1068  
`mllib_SignalLPCAutoCorrelGetEnergy_F32` — return the energy of the input signal, 1069  
`mllib_SignalLPCAutoCorrelGetEnergy_S16` — return the energy of the input signal, 1071  
`mllib_SignalLPCAutoCorrelGetEnergy_S16_Adap` — return the energy of the input signal, 1071  
`mllib_SignalLPCAutoCorrelGetPARCOR_F32` — return the partial correlation (PARCOR) coefficients, 1073  
`mllib_SignalLPCAutoCorrelGetPARCOR_S16` — return the partial correlation (PARCOR) coefficients, 1075  
`mllib_SignalLPCAutoCorrelGetPARCOR_S16_Adap` — return the partial correlation (PARCOR) coefficients, 1075  
`mllib_SignalLPCAutoCorrelInit_F32` — initialization for autocorrelation method of linear predictive coding, 1077  
`mllib_SignalLPCAutoCorrelInit_S16` — initialization for autocorrelation method of linear predictive coding, 1077



mlib\_SignalLPCCovariance\_F32 — perform linear predictive coding with covariance method, 1080  
 mlib\_SignalLPCCovariance\_S16 — perform linear predictive coding with covariance method, 1084  
 mlib\_SignalLPCCovariance\_S16\_Adap — perform linear predictive coding with covariance method, 1084  
 mlib\_SignalLPCCovarianceFree\_F32 — clean up for covariance method, 1082  
 mlib\_SignalLPCCovarianceFree\_S16 — clean up for covariance method, 1082  
 mlib\_SignalLPCCovarianceInit\_F32 — initialization for covariance method of linear predictive coding, 1083  
 mlib\_SignalLPCCovarianceInit\_S16 — initialization for covariance method of linear predictive coding, 1083  
 mlib\_SignalLPCPerceptWeight\_F32 — perform perceptual weighting on input signal, 1086  
 mlib\_SignalLPCPerceptWeight\_S16 — perform perceptual weighting on input signal, 1089  
 mlib\_SignalLPCPerceptWeightFree\_F32 — clean up for perceptual weighting, 1087  
 mlib\_SignalLPCPerceptWeightFree\_S16 — clean up for perceptual weighting, 1087  
 mlib\_SignalLPCPerceptWeightInit\_F32 — initialization for perceptual weighting of linear predictive coding, 1088  
 mlib\_SignalLPCPerceptWeightInit\_S16 — initialization for perceptual weighting of linear predictive coding, 1088  
 mlib\_SignalLPCPitchAnalyze\_F32 — perform open-loop pitch analysis, 1091  
 mlib\_SignalLPCPitchAnalyze\_S16 — perform open-loop pitch analysis, 1093  
 mlib\_SignalLSP2LPC\_F32 — convert line spectral pair coefficients to linear prediction coefficients, 1095  
 mlib\_SignalLSP2LPC\_S16 — convert line spectral pair coefficients to linear prediction coefficients, 1097  
 mlib\_SignalLSP2LPC\_S16\_Adap — convert line spectral pair coefficients to linear prediction coefficients, 1097  
 mlib\_SignalMelCepstral\_F32 — perform cepstral analysis in mel frequency scale, 1099  
 mlib\_SignalMelCepstral\_S16 — perform cepstral analysis in mel frequency scale, 1104  
 mlib\_SignalMelCepstral\_S16\_Adap — perform cepstral analysis in mel frequency scale, 1106  
 mlib\_SignalMelCepstralFree\_F32 — clean up for cepstral analysis in mel frequency scale, 1101  
 mlib\_SignalMelCepstralFree\_S16 — clean up for cepstral analysis in mel frequency scale, 1101  
 mlib\_SignalMelCepstralInit\_F32 — initialization for cepstral analysis in mel frequency scale, 1102  
 mlib\_SignalMelCepstralInit\_S16 — initialization for cepstral analysis in mel frequency scale, 1102  
 mlib\_SignalMerge\_F32S\_F32 — merge, 1108  
 mlib\_SignalMerge\_S16S\_S16 — merge, 1109  
 mlib\_SignalMul\_F32 — multiplication, 1118  
 mlib\_SignalMul\_F32\_F32 — multiplication, 1119  
 mlib\_SignalMul\_F32S — multiplication, 1118  
 mlib\_SignalMul\_F32S\_F32S — multiplication, 1119  
 mlib\_SignalMul\_S16\_S16\_Sat — multiplication, 1136  
 mlib\_SignalMul\_S16\_Sat — multiplication, 1137  
 mlib\_SignalMul\_S16S\_S16S\_Sat — multiplication, 1136  
 mlib\_SignalMul\_S16S\_Sat — multiplication, 1137  
 mlib\_SignalMulBartlett\_F32 — Bartlett windowing multiplication, 1110  
 mlib\_SignalMulBartlett\_F32\_F32 — Bartlett windowing multiplication, 1111  
 mlib\_SignalMulBartlett\_F32S — Bartlett windowing multiplication, 1110  
 mlib\_SignalMulBartlett\_F32S\_F32S — Bartlett windowing multiplication, 1111  
 mlib\_SignalMulBartlett\_S16 — Bartlett windowing multiplication, 1112

mlib\_SignalMulBartlett\_S16\_S16 — Bartlett windowing multiplication, 1113  
 mlib\_SignalMulBartlett\_S16S — Bartlett windowing multiplication, 1112  
 mlib\_SignalMulBartlett\_S16S\_S16S — Bartlett windowing multiplication, 1113  
 mlib\_SignalMulBlackman\_F32 — Blackman windowing multiplication, 1114  
 mlib\_SignalMulBlackman\_F32\_F32 — Blackman windowing multiplication, 1115  
 mlib\_SignalMulBlackman\_F32S — Blackman windowing multiplication, 1114  
 mlib\_SignalMulBlackman\_F32S\_F32S — Blackman windowing multiplication, 1115  
 mlib\_SignalMulBlackman\_S16 — Blackman windowing multiplication, 1116  
 mlib\_SignalMulBlackman\_S16\_S16 — Blackman windowing multiplication, 1117  
 mlib\_SignalMulBlackman\_S16S — Blackman windowing multiplication, 1116  
 mlib\_SignalMulBlackman\_S16S\_S16S — Blackman windowing multiplication, 1117  
 mlib\_SignalMulHamming\_F32 — Bartlett windowing multiplication, 1120  
 mlib\_SignalMulHamming\_F32\_F32 — Hamming windowing multiplication, 1121  
 mlib\_SignalMulHamming\_F32S — Bartlett windowing multiplication, 1120  
 mlib\_SignalMulHamming\_F32S\_F32S — Hamming windowing multiplication, 1121  
 mlib\_SignalMulHamming\_S16 — Bartlett windowing multiplication, 1122  
 mlib\_SignalMulHamming\_S16\_S16 — Hamming windowing multiplication, 1123  
 mlib\_SignalMulHamming\_S16S — Bartlett windowing multiplication, 1122  
 mlib\_SignalMulHamming\_S16S\_S16S — Hamming windowing multiplication, 1123  
 mlib\_SignalMulHanning\_F32 — Hanning windowing multiplication, 1124  
 mlib\_SignalMulHanning\_F32\_F32 — Hanning windowing multiplication, 1125  
 mlib\_SignalMulHanning\_F32S — Hanning windowing multiplication, 1124  
 mlib\_SignalMulHanning\_F32S\_F32S — Hanning windowing multiplication, 1125  
 mlib\_SignalMulHanning\_S16 — Hanning windowing multiplication, 1126  
 mlib\_SignalMulHanning\_S16\_S16 — Hanning windowing multiplication, 1127  
 mlib\_SignalMulHanning\_S16S — Hanning windowing multiplication, 1126  
 mlib\_SignalMulHanning\_S16S\_S16S — Hanning windowing multiplication, 1127  
 mlib\_SignalMulKaiser\_F32 — Kaiser windowing multiplication, 1128  
 mlib\_SignalMulKaiser\_F32\_F32 — Kaiser windowing multiplication, 1129  
 mlib\_SignalMulKaiser\_F32S — Kaiser windowing multiplication, 1128  
 mlib\_SignalMulKaiser\_F32S\_F32S — Kaiser windowing multiplication, 1129  
 mlib\_SignalMulKaiser\_S16 — Kaiser windowing multiplication, 1130  
 mlib\_SignalMulKaiser\_S16\_S16 — Kaiser windowing multiplication, 1131  
 mlib\_SignalMulKaiser\_S16S — Kaiser windowing multiplication, 1130  
 mlib\_SignalMulKaiser\_S16S\_S16S — Kaiser windowing multiplication, 1131  
 mlib\_SignalMulRectangular\_F32 — rectangular windowing multiplication, 1132  
 mlib\_SignalMulRectangular\_F32\_F32 — rectangular windowing multiplication, 1133  
 mlib\_SignalMulRectangular\_F32S — rectangular windowing multiplication, 1132  
 mlib\_SignalMulRectangular\_F32S\_F32S — rectangular windowing multiplication, 1133  
 mlib\_SignalMulRectangular\_S16 — rectangular windowing multiplication, 1134  
 mlib\_SignalMulRectangular\_S16\_S16 — rectangular windowing multiplication, 1135  
 mlib\_SignalMulRectangular\_S16S — rectangular windowing multiplication, 1134  
 mlib\_SignalMulRectangular\_S16S\_S16S — rectangular windowing multiplication, 1135  
 mlib\_SignalMulS\_F32 — multiplication by a scalar, 1142  
 mlib\_SignalMulS\_F32\_F32 — multiplication by a scalar, 1143  
 mlib\_SignalMulS\_F32S — multiplication by a scalar, 1142  
 mlib\_SignalMulS\_F32S\_F32S — multiplication by a scalar, 1143  
 mlib\_SignalMulS\_S16\_S16\_Sat — multiplication by a scalar, 1146

mlib\_SignalMulS\_S16\_Sat — multiplication by a scalar, 1147  
 mlib\_SignalMulS\_S16S\_S16S\_Sat — multiplication by a scalar, 1146  
 mlib\_SignalMulS\_S16S\_Sat — multiplication by a scalar, 1147  
 mlib\_SignalMulSAdd\_F32 — multiplication by a scalar plus addition, 1138  
 mlib\_SignalMulSAdd\_F32\_F32 — multiplication by a scalar plus addition, 1139  
 mlib\_SignalMulSAdd\_F32S — multiplication by a scalar plus addition, 1138  
 mlib\_SignalMulSAdd\_F32S\_F32S — multiplication by a scalar plus addition, 1139  
 mlib\_SignalMulSAdd\_S16\_S16\_Sat — multiplication by a scalar plus addition, 1140  
 mlib\_SignalMulSAdd\_S16\_Sat — multiplication by a scalar plus addition, 1141  
 mlib\_SignalMulSAdd\_S16S\_S16S\_Sat — multiplication by a scalar plus addition, 1140  
 mlib\_SignalMulSAdd\_S16S\_Sat — multiplication by a scalar plus addition, 1141  
 mlib\_SignalMulShift\_S16\_S16\_Sat — multiplication with shifting, 1144  
 mlib\_SignalMulShift\_S16\_Sat — multiplication with shifting, 1145  
 mlib\_SignalMulShift\_S16S\_S16S\_Sat — multiplication with shifting, 1144  
 mlib\_SignalMulShift\_S16S\_Sat — multiplication with shifting, 1145  
 mlib\_SignalMulSShift\_S16\_S16\_Sat — multiplication by a scalar with shifting, 1150  
 mlib\_SignalMulSShift\_S16\_Sat — multiplication by a scalar with shifting, 1151  
 mlib\_SignalMulSShift\_S16S\_S16S\_Sat — multiplication by a scalar with shifting, 1150  
 mlib\_SignalMulSShift\_S16S\_Sat — multiplication by a scalar with shifting, 1151  
 mlib\_SignalMulSShiftAdd\_S16\_S16\_Sat — multiplication by a scalar plus addition, 1148  
 mlib\_SignalMulSShiftAdd\_S16\_Sat — multiplication by a scalar plus addition, 1149  
 mlib\_SignalMulSShiftAdd\_S16S\_S16S\_Sat — multiplication by a scalar plus addition, 1148  
 mlib\_SignalMulSShiftAdd\_S16S\_Sat — multiplication by a scalar plus addition, 1149  
 mlib\_SignalMulWindow\_F32 — windowing, 1152  
 mlib\_SignalMulWindow\_F32\_F32 — windowing, 1153  
 mlib\_SignalMulWindow\_F32S — windowing, 1154  
 mlib\_SignalMulWindow\_F32S\_F32S — windowing, 1155  
 mlib\_SignalMulWindow\_S16 — windowing, 1156  
 mlib\_SignalMulWindow\_S16\_S16 — windowing, 1157  
 mlib\_SignalMulWindow\_S16S — windowing, 1156  
 mlib\_SignalMulWindow\_S16S\_S16S — windowing, 1157  
 mlib\_SignalQuant\_S16\_F32 — float to 16-bit quantization, 1160  
 mlib\_SignalQuant\_S16S\_F32S — float to 16-bit quantization, 1161  
 mlib\_SignalQuant\_U8\_F32 — float to 8-bit quantization, 1162  
 mlib\_SignalQuant\_U8\_S16 — 16-bit to 8-bit quantization, 1163  
 mlib\_SignalQuant\_U8S\_F32S — float to 8-bit quantization, 1164  
 mlib\_SignalQuant\_U8S\_S16S — 16-bit to 8-bit quantization, 1163  
 mlib\_SignalQuant2\_S16\_F32 — float to 16-bit quantization, 1158  
 mlib\_SignalQuant2\_S16S\_F32S — float to 16-bit quantization, 1159  
 mlib\_SignalReSampleFIR\_F32\_F32 — resampling with filtering, 1165  
 mlib\_SignalReSampleFIR\_F32S\_F32S — resampling with filtering, 1166  
 mlib\_SignalReSampleFIR\_S16\_S16\_Sat — resampling with filtering, 1172

mlib\_SignalReSampleFIR\_S16S\_S16S\_Sat — resampling with filtering, 1172  
 mlib\_SignalReSampleFIRFree\_F32\_F32 — resampling with filtering, 1167  
 mlib\_SignalReSampleFIRFree\_F32S\_F32S — resampling with filtering, 1168  
 mlib\_SignalReSampleFIRFree\_S16\_S16 — resampling with filtering, 1169  
 mlib\_SignalReSampleFIRFree\_S16S\_S16S — resampling with filtering, 1169  
 mlib\_SignalReSampleFIRInit\_F32\_F32 — initialization for resampling with filtering, 1170  
 mlib\_SignalReSampleFIRInit\_F32S\_F32S — initialization for resampling with filtering, 1170  
 mlib\_SignalReSampleFIRInit\_S16\_S16 — initialization for resampling with filtering, 1170  
 mlib\_SignalReSampleFIRInit\_S16S\_S16S — initialization for resampling with filtering, 1170  
 mlib\_SignalSineWave\_F32 — sine wave generation, 1173  
 mlib\_SignalSineWave\_S16 — sine wave generation, 1178  
 mlib\_SignalSineWaveFree\_F32 — sine wave generation, 1174  
 mlib\_SignalSineWaveFree\_S16 — sine wave generation, 1175  
 mlib\_SignalSineWaveInit\_F32 — sine wave generation, 1176  
 mlib\_SignalSineWaveInit\_S16 — sine wave generation, 1177  
 mlib\_SignalSplit\_F32\_F32S — split, 1179  
 mlib\_SignalSplit\_S16\_S16S — split, 1180  
 mlib\_SignaluLaw2ALaw — ITU G.711 m-law and A-law compression and decompression, 1181  
 mlib\_SignaluLaw2Linear — ITU G.711 m-law and A-law compression and decompression, 1182  
 mlib\_SignalUpSample\_F32\_F32 — signal upsampling, 1192  
 mlib\_SignalUpSample\_F32S\_F32S — signal upsampling, 1192  
 mlib\_SignalUpSample\_S16\_S16 — signal upsampling, 1192  
 mlib\_SignalUpSample\_S16S\_S16S — signal upsampling, 1192  
 mlib\_SignalUpSampleFIR\_F32\_F32 — upsampling with filtering, 1183  
 mlib\_SignalUpSampleFIR\_F32S\_F32S — upsampling with filtering, 1184  
 mlib\_SignalUpSampleFIR\_S16\_S16\_Sat — upsampling with filtering, 1191  
 mlib\_SignalUpSampleFIR\_S16S\_S16S\_Sat — upsampling with filtering, 1191  
 mlib\_SignalUpSampleFIRFree\_F32\_F32 — upsampling with filtering, 1185  
 mlib\_SignalUpSampleFIRFree\_F32S\_F32S — upsampling with filtering, 1186  
 mlib\_SignalUpSampleFIRFree\_S16\_S16 — upsampling with filtering, 1187  
 mlib\_SignalUpSampleFIRFree\_S16S\_S16S — upsampling with filtering, 1187  
 mlib\_SignalUpSampleFIRInit\_F32\_F32 — upsampling with filtering, 1188  
 mlib\_SignalUpSampleFIRInit\_F32S\_F32S — upsampling with filtering, 1189  
 mlib\_SignalUpSampleFIRInit\_S16\_S16 — upsampling with filtering, 1190  
 mlib\_SignalUpSampleFIRInit\_S16S\_S16S — upsampling with filtering, 1190  
 mlib\_SignalWhiteNoise\_F32 — white noise generation, 1194  
 mlib\_SignalWhiteNoise\_S16 — white noise generation, 1199  
 mlib\_SignalWhiteNoiseFree\_F32 — white noise generation, 1195  
 mlib\_SignalWhiteNoiseFree\_S16 — white noise generation, 1196  
 mlib\_SignalWhiteNoiseInit\_F32 — white noise generation, 1197  
 mlib\_SignalWhiteNoiseInit\_S16 — white noise generation, 1198  
 mlib\_VectorAdd\_S16\_Mod — vector addition, in place, 1205  
 mlib\_VectorAdd\_S16\_S16\_Mod — vector addition, 1207  
 mlib\_VectorAdd\_S16\_S16\_Sat — vector addition, 1207  
 mlib\_VectorAdd\_S16\_S8\_Mod — vector addition, 1207  
 mlib\_VectorAdd\_S16\_S8\_Sat — vector addition, 1207

mlib\_VectorAdd\_S16\_Sat — vector addition, in place, 1205  
 mlib\_VectorAdd\_S16\_U8\_Mod — vector addition, 1207  
 mlib\_VectorAdd\_S16\_U8\_Sat — vector addition, 1207  
 mlib\_VectorAdd\_S16C\_Mod — vector addition, in place, 1205  
 mlib\_VectorAdd\_S16C\_S16C\_Mod — vector addition, 1207  
 mlib\_VectorAdd\_S16C\_S16C\_Sat — vector addition, 1207  
 mlib\_VectorAdd\_S16C\_S8C\_Mod — vector addition, 1207  
 mlib\_VectorAdd\_S16C\_S8C\_Sat — vector addition, 1207  
 mlib\_VectorAdd\_S16C\_Sat — vector addition, in place, 1205  
 mlib\_VectorAdd\_S16C\_U8C\_Mod — vector addition, 1207  
 mlib\_VectorAdd\_S16C\_U8C\_Sat — vector addition, 1207  
 mlib\_VectorAdd\_S32\_Mod — vector addition, in place, 1205  
 mlib\_VectorAdd\_S32\_S16\_Mod — vector addition, 1207  
 mlib\_VectorAdd\_S32\_S16\_Sat — vector addition, 1207  
 mlib\_VectorAdd\_S32\_S32\_Mod — vector addition, 1207  
 mlib\_VectorAdd\_S32\_S32\_Sat — vector addition, 1207  
 mlib\_VectorAdd\_S32\_Sat — vector addition, in place, 1205  
 mlib\_VectorAdd\_S32C\_Mod — vector addition, in place, 1205  
 mlib\_VectorAdd\_S32C\_S16C\_Mod — vector addition, 1207  
 mlib\_VectorAdd\_S32C\_S16C\_Sat — vector addition, 1207  
 mlib\_VectorAdd\_S32C\_S32C\_Mod — vector addition, 1207  
 mlib\_VectorAdd\_S32C\_S32C\_Sat — vector addition, 1207  
 mlib\_VectorAdd\_S32C\_Sat — vector addition, in place, 1205  
 mlib\_VectorAdd\_S8\_Mod — vector addition, in place, 1205  
 mlib\_VectorAdd\_S8\_S8\_Mod — vector addition, 1207  
 mlib\_VectorAdd\_S8\_S8\_Sat — vector addition, 1207  
 mlib\_VectorAdd\_S8\_Sat — vector addition, in place, 1205  
 mlib\_VectorAdd\_S8C\_Mod — vector addition, in place, 1205  
 mlib\_VectorAdd\_S8C\_S8C\_Mod — vector addition, 1207  
 mlib\_VectorAdd\_S8C\_S8C\_Sat — vector addition, 1207  
 mlib\_VectorAdd\_S8C\_Sat — vector addition, in place, 1205  
 mlib\_VectorAdd\_U8\_Mod — vector addition, in place, 1205  
 mlib\_VectorAdd\_U8\_Sat — vector addition, in place, 1205  
 mlib\_VectorAdd\_U8\_U8\_Mod — vector addition, 1207  
 mlib\_VectorAdd\_U8\_U8\_Sat — vector addition, 1207  
 mlib\_VectorAdd\_U8C\_Mod — vector addition, in place, 1205  
 mlib\_VectorAdd\_U8C\_Sat — vector addition, in place, 1205  
 mlib\_VectorAdd\_U8C\_U8C\_Mod — vector addition, 1207  
 mlib\_VectorAdd\_U8C\_U8C\_Sat — vector addition, 1207  
 mlib\_VectorAddS\_S16\_Mod — vector addition to scalar, in place, 1200  
 mlib\_VectorAddS\_S16\_S16\_Mod — vector addition to scalar, 1202  
 mlib\_VectorAddS\_S16\_S16\_Sat — vector addition to scalar, 1202  
 mlib\_VectorAddS\_S16\_S8\_Mod — vector addition to scalar, 1202  
 mlib\_VectorAddS\_S16\_S8\_Sat — vector addition to scalar, 1202  
 mlib\_VectorAddS\_S16\_Sat — vector addition to scalar, in place, 1200  
 mlib\_VectorAddS\_S16\_U8\_Mod — vector addition to scalar, 1202  
 mlib\_VectorAddS\_S16\_U8\_Sat — vector addition to scalar, 1202  
 mlib\_VectorAddS\_S16C\_Mod — vector addition to scalar, in place, 1200

mlib\_VectorAddS\_S16C\_S16C\_Mod — vector addition to scalar, 1202  
 mlib\_VectorAddS\_S16C\_S16C\_Sat — vector addition to scalar, 1202  
 mlib\_VectorAddS\_S16C\_S8C\_Mod — vector addition to scalar, 1202  
 mlib\_VectorAddS\_S16C\_S8C\_Sat — vector addition to scalar, 1202  
 mlib\_VectorAddS\_S16C\_Sat — vector addition to scalar, in place, 1200  
 mlib\_VectorAddS\_S16C\_U8C\_Mod — vector addition to scalar, 1202  
 mlib\_VectorAddS\_S16C\_U8C\_Sat — vector addition to scalar, 1202  
 mlib\_VectorAddS\_S32\_Mod — vector addition to scalar, in place, 1200  
 mlib\_VectorAddS\_S32\_S16\_Mod — vector addition to scalar, 1202  
 mlib\_VectorAddS\_S32\_S16\_Sat — vector addition to scalar, 1202  
 mlib\_VectorAddS\_S32\_S32\_Mod — vector addition to scalar, 1202  
 mlib\_VectorAddS\_S32\_S32\_Sat — vector addition to scalar, 1202  
 mlib\_VectorAddS\_S32\_Sat — vector addition to scalar, in place, 1200  
 mlib\_VectorAddS\_S32C\_Mod — vector addition to scalar, in place, 1200  
 mlib\_VectorAddS\_S32C\_S16C\_Mod — vector addition to scalar, 1202  
 mlib\_VectorAddS\_S32C\_S16C\_Sat — vector addition to scalar, 1202  
 mlib\_VectorAddS\_S32C\_S32C\_Mod — vector addition to scalar, 1202  
 mlib\_VectorAddS\_S32C\_S32C\_Sat — vector addition to scalar, 1202  
 mlib\_VectorAddS\_S32C\_Sat — vector addition to scalar, in place, 1200  
 mlib\_VectorAddS\_S8\_Mod — vector addition to scalar, in place, 1200  
 mlib\_VectorAddS\_S8\_S8\_Mod — vector addition to scalar, 1202  
 mlib\_VectorAddS\_S8\_S8\_Sat — vector addition to scalar, 1202  
 mlib\_VectorAddS\_S8\_Sat — vector addition to scalar, in place, 1200  
 mlib\_VectorAddS\_S8C\_Mod — vector addition to scalar, in place, 1200  
 mlib\_VectorAddS\_S8C\_S8C\_Mod — vector addition to scalar, 1202  
 mlib\_VectorAddS\_S8C\_S8C\_Sat — vector addition to scalar, 1202  
 mlib\_VectorAddS\_S8C\_Sat — vector addition to scalar, in place, 1200  
 mlib\_VectorAddS\_U8\_Mod — vector addition to scalar, in place, 1200  
 mlib\_VectorAddS\_U8\_Sat — vector addition to scalar, in place, 1200  
 mlib\_VectorAddS\_U8\_U8\_Mod — vector addition to scalar, 1202  
 mlib\_VectorAddS\_U8\_U8\_Sat — vector addition to scalar, 1202  
 mlib\_VectorAddS\_U8C\_Mod — vector addition to scalar, in place, 1200  
 mlib\_VectorAddS\_U8C\_Sat — vector addition to scalar, in place, 1200  
 mlib\_VectorAddS\_U8C\_U8C\_Mod — vector addition to scalar, 1202  
 mlib\_VectorAddS\_U8C\_U8C\_Sat — vector addition to scalar, 1202  
 mlib\_VectorAng\_S16C — vector complex phase (angle), 1210  
 mlib\_VectorAng\_S32C — vector complex phase (angle), 1210  
 mlib\_VectorAng\_S8C — vector complex phase (angle), 1210  
 mlib\_VectorAng\_U8C — vector complex phase (angle), 1210  
 mlib\_VectorConj\_S16C\_S16C\_Sat — vector conjugation, 1212  
 mlib\_VectorConj\_S16C\_Sat — vector conjugation, 1213  
 mlib\_VectorConj\_S32C\_S32C\_Sat — vector conjugation, 1212  
 mlib\_VectorConj\_S32C\_Sat — vector conjugation, 1213  
 mlib\_VectorConj\_S8C\_S8C\_Sat — vector conjugation, 1212  
 mlib\_VectorConj\_S8C\_Sat — vector conjugation, 1213  
 mlib\_VectorConjRev\_S16C\_S16C\_Sat — vector conjugation reversion, 1211  
 mlib\_VectorConjRev\_S32C\_S32C\_Sat — vector conjugation reversion, 1211  
 mlib\_VectorConjRev\_S8C\_S8C\_Sat — vector conjugation reversion, 1211

mlib\_VectorConjSymExt\_S16C\_S16C\_Sat — vector conjugate-symmetric extension, 1214  
 mlib\_VectorConjSymExt\_S32C\_S32C\_Sat — vector conjugate-symmetric extension, 1214  
 mlib\_VectorConjSymExt\_S8C\_S8C\_Sat — vector conjugate-symmetric extension, 1214  
 mlib\_VectorConvert\_S16\_S32\_Mod — vector data type convert, 1216  
 mlib\_VectorConvert\_S16\_S32\_Sat — vector data type convert, 1216  
 mlib\_VectorConvert\_S16\_S8\_Mod — vector data type convert, 1216  
 mlib\_VectorConvert\_S16\_S8\_Sat — vector data type convert, 1216  
 mlib\_VectorConvert\_S16\_U8\_Mod — vector data type convert, 1216  
 mlib\_VectorConvert\_S16\_U8\_Sat — vector data type convert, 1216  
 mlib\_VectorConvert\_S16C\_S32C\_Mod — vector data type convert, 1216  
 mlib\_VectorConvert\_S16C\_S32C\_Sat — vector data type convert, 1216  
 mlib\_VectorConvert\_S16C\_S8C\_Mod — vector data type convert, 1216  
 mlib\_VectorConvert\_S16C\_S8C\_Sat — vector data type convert, 1216  
 mlib\_VectorConvert\_S16C\_U8C\_Mod — vector data type convert, 1216  
 mlib\_VectorConvert\_S16C\_U8C\_Sat — vector data type convert, 1216  
 mlib\_VectorConvert\_S32\_S16\_Mod — vector data type convert, 1216  
 mlib\_VectorConvert\_S32\_S16\_Sat — vector data type convert, 1216  
 mlib\_VectorConvert\_S32\_S8\_Mod — vector data type convert, 1216  
 mlib\_VectorConvert\_S32\_S8\_Sat — vector data type convert, 1216  
 mlib\_VectorConvert\_S32\_U8\_Mod — vector data type convert, 1216  
 mlib\_VectorConvert\_S32\_U8\_Sat — vector data type convert, 1216  
 mlib\_VectorConvert\_S32C\_S16C\_Mod — vector data type convert, 1216  
 mlib\_VectorConvert\_S32C\_S16C\_Sat — vector data type convert, 1216  
 mlib\_VectorConvert\_S32C\_S8C\_Mod — vector data type convert, 1216  
 mlib\_VectorConvert\_S32C\_S8C\_Sat — vector data type convert, 1216  
 mlib\_VectorConvert\_S32C\_U8C\_Mod — vector data type convert, 1216  
 mlib\_VectorConvert\_S32C\_U8C\_Sat — vector data type convert, 1216  
 mlib\_VectorConvert\_S8\_S16\_Mod — vector data type convert, 1216  
 mlib\_VectorConvert\_S8\_S16\_Sat — vector data type convert, 1216  
 mlib\_VectorConvert\_S8\_S32\_Mod — vector data type convert, 1216  
 mlib\_VectorConvert\_S8\_S32\_Sat — vector data type convert, 1216  
 mlib\_VectorConvert\_S8\_S8\_Mod — vector data type convert, 1216  
 mlib\_VectorConvert\_S8\_S8\_Sat — vector data type convert, 1216  
 mlib\_VectorConvert\_S8\_U8\_Mod — vector data type convert, 1216  
 mlib\_VectorConvert\_S8\_U8\_Sat — vector data type convert, 1216  
 mlib\_VectorConvert\_S8C\_S16C\_Mod — vector data type convert, 1216  
 mlib\_VectorConvert\_S8C\_S16C\_Sat — vector data type convert, 1216  
 mlib\_VectorConvert\_S8C\_S32C\_Mod — vector data type convert, 1216  
 mlib\_VectorConvert\_S8C\_S32C\_Sat — vector data type convert, 1216  
 mlib\_VectorConvert\_S8C\_U8C\_Mod — vector data type convert, 1216  
 mlib\_VectorConvert\_S8C\_U8C\_Sat — vector data type convert, 1216  
 mlib\_VectorConvert\_U8\_S16\_Mod — vector data type convert, 1216  
 mlib\_VectorConvert\_U8\_S16\_Sat — vector data type convert, 1216  
 mlib\_VectorConvert\_U8\_S32\_Mod — vector data type convert, 1216  
 mlib\_VectorConvert\_U8\_S32\_Sat — vector data type convert, 1216  
 mlib\_VectorConvert\_U8\_S8\_Mod — vector data type convert, 1216  
 mlib\_VectorConvert\_U8\_S8\_Sat — vector data type convert, 1216  
 mlib\_VectorConvert\_U8C\_S16C\_Mod — vector data type convert, 1216  
 mlib\_VectorConvert\_U8C\_S16C\_Sat — vector data type convert, 1216  
 mlib\_VectorConvert\_U8C\_S32C\_Mod — vector data type convert, 1216  
 mlib\_VectorConvert\_U8C\_S32C\_Sat — vector data type convert, 1216

mlib\_VectorConvert\_U8C\_S32C\_Sat — vector data type convert, 1216  
 mlib\_VectorConvert\_U8C\_S8C\_Mod — vector data type convert, 1216  
 mlib\_VectorConvert\_U8C\_S8C\_Sat — vector data type convert, 1216  
 mlib\_VectorCopy\_S16 — vector copy, 1221  
 mlib\_VectorCopy\_S16C — vector copy, 1221  
 mlib\_VectorCopy\_S32 — vector copy, 1221  
 mlib\_VectorCopy\_S32C — vector copy, 1221  
 mlib\_VectorCopy\_S8 — vector copy, 1221  
 mlib\_VectorCopy\_S8C — vector copy, 1221  
 mlib\_VectorCopy\_U8 — vector copy, 1221  
 mlib\_VectorCopy\_U8C — vector copy, 1221  
 mlib\_VectorDistance\_S16\_Sat — vector Euclidean distance, 1223  
 mlib\_VectorDistance\_S32\_Sat — vector Euclidean distance, 1223  
 mlib\_VectorDistance\_S8\_Sat — vector Euclidean distance, 1223  
 mlib\_VectorDistance\_U8\_Sat — vector Euclidean distance, 1223  
 mlib\_VectorDotProd\_S16\_Sat — vector dot product (inner product), 1224  
 mlib\_VectorDotProd\_S16C\_Sat — vector dot product (inner product), 1224  
 mlib\_VectorDotProd\_S32\_Sat — vector dot product (inner product), 1224  
 mlib\_VectorDotProd\_S32C\_Sat — vector dot product (inner product), 1224  
 mlib\_VectorDotProd\_S8\_Sat — vector dot product (inner product), 1224  
 mlib\_VectorDotProd\_S8C\_Sat — vector dot product (inner product), 1224  
 mlib\_VectorDotProd\_U8\_Sat — vector dot product (inner product), 1224  
 mlib\_VectorDotProd\_U8C\_Sat — vector dot product (inner product), 1224  
 mlib\_VectorMag\_S16C — vector complex magnitude, 1226  
 mlib\_VectorMag\_S32C — vector complex magnitude, 1226  
 mlib\_VectorMag\_S8C — vector complex magnitude, 1226  
 mlib\_VectorMag\_U8C — vector complex magnitude, 1226  
 mlib\_VectorMaximum\_D64 — find the maximum value in a vector, 1228  
 mlib\_VectorMaximum\_F32 — find the maximum value in a vector, 1228  
 mlib\_VectorMaximum\_S16 — find the maximum value in a vector, 1228  
 mlib\_VectorMaximum\_S32 — find the maximum value in a vector, 1228  
 mlib\_VectorMaximum\_S8 — find the maximum value in a vector, 1228  
 mlib\_VectorMaximum\_U8 — find the maximum value in a vector, 1228  
 mlib\_VectorMaximumMag\_D64C — find the first element with the maximum magnitude in a vector, 1227  
 mlib\_VectorMaximumMag\_F32C — find the first element with the maximum magnitude in a vector, 1227  
 mlib\_VectorMaximumMag\_S16C — find the first element with the maximum magnitude in a vector, 1227  
 mlib\_VectorMaximumMag\_S32C — find the first element with the maximum magnitude in a vector, 1227  
 mlib\_VectorMaximumMag\_S8C — find the first element with the maximum magnitude in a vector, 1227  
 mlib\_VectorMaximumMag\_U8C — find the first element with the maximum magnitude in a vector, 1227  
 mlib\_VectorMerge\_S16C\_S16 — vector merge, 1229  
 mlib\_VectorMerge\_S32C\_S32 — vector merge, 1229  
 mlib\_VectorMerge\_S8C\_S8 — vector merge, 1229  
 mlib\_VectorMerge\_U8C\_U8 — vector merge, 1229  
 mlib\_VectorMinimum\_D64 — find the minimum value in a vector, 1231  
 mlib\_VectorMinimum\_F32 — find the minimum value in a vector, 1231  
 mlib\_VectorMinimum\_S16 — find the minimum value in a vector, 1231  
 mlib\_VectorMinimum\_S32 — find the minimum value in a vector, 1231  
 mlib\_VectorMinimum\_S8 — find the minimum value in a vector, 1231  
 mlib\_VectorMinimum\_U8 — find the minimum value in a vector, 1231



mlib\_VectorMinimumMag\_D64C — find the first element with the minimum magnitude in a vector, 1230  
 mlib\_VectorMinimumMag\_F32C — find the first element with the minimum magnitude in a vector, 1230  
 mlib\_VectorMinimumMag\_S16C — find the first element with the minimum magnitude in a vector, 1230  
 mlib\_VectorMinimumMag\_S32C — find the first element with the minimum magnitude in a vector, 1230  
 mlib\_VectorMinimumMag\_S8C — find the first element with the minimum magnitude in a vector, 1230  
 mlib\_VectorMinimumMag\_U8C — find the first element with the minimum magnitude in a vector, 1230  
 mlib\_VectorMul\_S16\_Mod — vector multiplication, in place, 1255  
 mlib\_VectorMul\_S16\_S16\_Mod — vector multiplication, 1257  
 mlib\_VectorMul\_S16\_S16\_Sat — vector multiplication, 1257  
 mlib\_VectorMul\_S16\_S8\_Mod — vector multiplication, 1257  
 mlib\_VectorMul\_S16\_S8\_Sat — vector multiplication, 1257  
 mlib\_VectorMul\_S16\_Sat — vector multiplication, in place, 1255  
 mlib\_VectorMul\_S16\_U8\_Mod — vector multiplication, 1257  
 mlib\_VectorMul\_S16\_U8\_Sat — vector multiplication, 1257  
 mlib\_VectorMul\_S16C\_Mod — vector multiplication, in place, 1255  
 mlib\_VectorMul\_S16C\_S16C\_Mod — vector multiplication, 1257  
 mlib\_VectorMul\_S16C\_S16C\_Sat — vector multiplication, 1257  
 mlib\_VectorMul\_S16C\_S8C\_Mod — vector multiplication, 1257  
 mlib\_VectorMul\_S16C\_S8C\_Sat — vector multiplication, 1257  
 mlib\_VectorMul\_S16C\_Sat — vector multiplication, in place, 1255  
 mlib\_VectorMul\_S16C\_U8C\_Mod — vector multiplication, 1257  
 mlib\_VectorMul\_S16C\_U8C\_Sat — vector multiplication, 1257  
 mlib\_VectorMul\_S32\_Mod — vector multiplication, in place, 1255  
 mlib\_VectorMul\_S32\_S16\_Mod — vector multiplication, 1257  
 mlib\_VectorMul\_S32\_S16\_Sat — vector multiplication, 1257  
 mlib\_VectorMul\_S32\_S32\_Mod — vector multiplication, 1257  
 mlib\_VectorMul\_S32\_S32\_Sat — vector multiplication, 1257  
 mlib\_VectorMul\_S32\_Sat — vector multiplication, in place, 1255  
 mlib\_VectorMul\_S32C\_Mod — vector multiplication, in place, 1255  
 mlib\_VectorMul\_S32C\_S16C\_Mod — vector multiplication, 1257  
 mlib\_VectorMul\_S32C\_S16C\_Sat — vector multiplication, 1257  
 mlib\_VectorMul\_S32C\_S32C\_Mod — vector multiplication, 1257  
 mlib\_VectorMul\_S32C\_S32C\_Sat — vector multiplication, 1257  
 mlib\_VectorMul\_S32C\_Sat — vector multiplication, in place, 1255  
 mlib\_VectorMul\_S8\_Mod — vector multiplication, in place, 1255  
 mlib\_VectorMul\_S8\_S8\_Mod — vector multiplication, 1257  
 mlib\_VectorMul\_S8\_S8\_Sat — vector multiplication, 1257  
 mlib\_VectorMul\_S8\_Sat — vector multiplication, in place, 1255  
 mlib\_VectorMul\_S8C\_Mod — vector multiplication, in place, 1255  
 mlib\_VectorMul\_S8C\_S8C\_Mod — vector multiplication, 1257  
 mlib\_VectorMul\_S8C\_S8C\_Sat — vector multiplication, 1257  
 mlib\_VectorMul\_S8C\_Sat — vector multiplication, in place, 1255  
 mlib\_VectorMul\_U8\_Mod — vector multiplication, in place, 1255  
 mlib\_VectorMul\_U8\_Sat — vector multiplication, in place, 1255  
 mlib\_VectorMul\_U8\_U8\_Mod — vector multiplication, 1257

mlib\_VectorMul\_U8\_U8\_Sat — vector multiplication, 1257  
 mlib\_VectorMul\_U8C\_Mod — vector multiplication, in place, 1255  
 mlib\_VectorMul\_U8C\_Sat — vector multiplication, in place, 1255  
 mlib\_VectorMul\_U8C\_U8C\_Mod — vector multiplication, 1257  
 mlib\_VectorMul\_U8C\_U8C\_Sat — vector multiplication, 1257  
 mlib\_VectorMulM\_S16\_S16\_Mod — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_S16\_S16\_Sat — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_S16\_S8\_Mod — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_S16\_S8\_Sat — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_S16\_U8\_Mod — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_S16\_U8\_Sat — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_S16C\_S16C\_Mod — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_S16C\_S16C\_Sat — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_S16C\_S8C\_Mod — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_S16C\_S8C\_Sat — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_S16C\_U8C\_Mod — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_S16C\_U8C\_Sat — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_S32\_S16\_Mod — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_S32\_S16\_Sat — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_S32\_S32\_Mod — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_S32\_S32\_Sat — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_S32C\_S16C\_Mod — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_S32C\_S16C\_Sat — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_S32C\_S32C\_Mod — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_S32C\_S32C\_Sat — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_S8\_S8\_Mod — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_S8\_S8\_Sat — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_S8C\_S8C\_Mod — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_S8C\_S8C\_Sat — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_U8\_U8\_Mod — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_U8\_U8\_Sat — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_U8C\_U8C\_Mod — multiplication of vector by matrix, 1234  
 mlib\_VectorMulM\_U8C\_U8C\_Sat — multiplication of vector by matrix, 1234  
 mlib\_VectorMulMShift\_S16\_S16\_Mod — multiplication of vector by matrix with shifting, 1232  
 mlib\_VectorMulMShift\_S16\_S16\_Sat — multiplication of vector by matrix with shifting, 1232  
 mlib\_VectorMulMShift\_S16C\_S16C\_Mod — multiplication of vector by matrix with shifting, 1232  
 mlib\_VectorMulMShift\_S16C\_S16C\_Sat — multiplication of vector by matrix with shifting, 1232  
 mlib\_VectorMulS\_S16\_Mod — vector multiplication by scalar, in place, 1250  
 mlib\_VectorMulS\_S16\_S16\_Mod — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_S16\_S16\_Sat — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_S16\_S8\_Mod — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_S16\_S8\_Sat — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_S16\_Sat — vector multiplication by scalar, in place, 1250  
 mlib\_VectorMulS\_S16\_U8\_Mod — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_S16\_U8\_Sat — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_S16C\_Mod — vector multiplication by scalar, in place, 1250

mlib\_VectorMulS\_S16C\_S16C\_Mod — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_S16C\_S16C\_Sat — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_S16C\_S8C\_Mod — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_S16C\_S8C\_Sat — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_S16C\_Sat — vector multiplication by scalar, in place, 1250  
 mlib\_VectorMulS\_S16C\_U8C\_Mod — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_S16C\_U8C\_Sat — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_S32\_Mod — vector multiplication by scalar, in place, 1250  
 mlib\_VectorMulS\_S32\_S16\_Mod — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_S32\_S16\_Sat — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_S32\_S32\_Mod — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_S32\_S32\_Sat — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_S32\_Sat — vector multiplication by scalar, in place, 1250  
 mlib\_VectorMulS\_S32C\_Mod — vector multiplication by scalar, in place, 1250  
 mlib\_VectorMulS\_S32C\_S16C\_Mod — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_S32C\_S16C\_Sat — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_S32C\_S32C\_Mod — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_S32C\_S32C\_Sat — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_S32C\_Sat — vector multiplication by scalar, in place, 1250  
 mlib\_VectorMulS\_S8\_Mod — vector multiplication by scalar, in place, 1250  
 mlib\_VectorMulS\_S8\_S8\_Mod — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_S8\_S8\_Sat — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_S8\_Sat — vector multiplication by scalar, in place, 1250  
 mlib\_VectorMulS\_S8C\_Mod — vector multiplication by scalar, in place, 1250  
 mlib\_VectorMulS\_S8C\_S8C\_Mod — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_S8C\_S8C\_Sat — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_S8C\_Sat — vector multiplication by scalar, in place, 1250  
 mlib\_VectorMulS\_U8\_Mod — vector multiplication by scalar, in place, 1250  
 mlib\_VectorMulS\_U8\_Sat — vector multiplication by scalar, in place, 1250  
 mlib\_VectorMulS\_U8\_U8\_Mod — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_U8\_U8\_Sat — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_U8C\_Mod — vector multiplication by scalar, in place, 1250  
 mlib\_VectorMulS\_U8C\_Sat — vector multiplication by scalar, in place, 1250  
 mlib\_VectorMulS\_U8C\_U8C\_Mod — vector multiplication by scalar, 1252  
 mlib\_VectorMulS\_U8C\_U8C\_Sat — vector multiplication by scalar, 1252  
 mlib\_VectorMulSAdd\_S16\_Mod — vector multiplication by scalar plus addition, in place, 1237  
 mlib\_VectorMulSAdd\_S16\_S16\_Mod — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulSAdd\_S16\_S16\_Sat — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulSAdd\_S16\_S8\_Mod — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulSAdd\_S16\_S8\_Sat — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulSAdd\_S16\_Sat — vector multiplication by scalar plus addition, in place, 1237  
 mlib\_VectorMulSAdd\_S16\_U8\_Mod — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulSAdd\_S16\_U8\_Sat — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulSAdd\_S16C\_Mod — vector multiplication by scalar plus addition, in place, 1237  
 mlib\_VectorMulSAdd\_S16C\_S16C\_Mod — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulSAdd\_S16C\_S16C\_Sat — vector multiplication by scalar plus addition, 1239

mlib\_VectorMulSAdd\_S16C\_S8C\_Mod — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulSAdd\_S16C\_S8C\_Sat — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulSAdd\_S16C\_Sat — vector multiplication by scalar plus addition, in place, 1237  
 mlib\_VectorMulSAdd\_S16C\_U8C\_Mod — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulSAdd\_S16C\_U8C\_Sat — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulSAdd\_S32\_Mod — vector multiplication by scalar plus addition, in place, 1237  
 mlib\_VectorMulSAdd\_S32\_S16\_Mod — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulSAdd\_S32\_S16\_Sat — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulSAdd\_S32\_S32\_Mod — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulSAdd\_S32\_S32\_Sat — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulSAdd\_S32\_Sat — vector multiplication by scalar plus addition, in place, 1237  
 mlib\_VectorMulSAdd\_S32C\_Mod — vector multiplication by scalar plus addition, in place, 1237  
 mlib\_VectorMulSAdd\_S32C\_S16C\_Mod — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulSAdd\_S32C\_S16C\_Sat — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulSAdd\_S32C\_S32C\_Mod — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulSAdd\_S32C\_S32C\_Sat — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulSAdd\_S32C\_Sat — vector multiplication by scalar plus addition, in place, 1237  
 mlib\_VectorMulSAdd\_S8\_Mod — vector multiplication by scalar plus addition, in place, 1237  
 mlib\_VectorMulSAdd\_S8\_S8\_Mod — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulSAdd\_S8\_S8\_Sat — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulSAdd\_S8\_Sat — vector multiplication by scalar plus addition, in place, 1237  
 mlib\_VectorMulSAdd\_S8C\_Mod — vector multiplication by scalar plus addition, in place, 1237  
 mlib\_VectorMulSAdd\_S8C\_S8C\_Mod — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulSAdd\_S8C\_S8C\_Sat — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulSAdd\_S8C\_Sat — vector multiplication by scalar plus addition, in place, 1237  
 mlib\_VectorMulSAdd\_U8\_Mod — vector multiplication by scalar plus addition, in place, 1237  
 mlib\_VectorMulSAdd\_U8\_Sat — vector multiplication by scalar plus addition, in place, 1237  
 mlib\_VectorMulSAdd\_U8\_U8\_Mod — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulSAdd\_U8\_U8\_Sat — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulSAdd\_U8C\_Mod — vector multiplication by scalar plus addition, in place, 1237  
 mlib\_VectorMulSAdd\_U8C\_Sat — vector multiplication by scalar plus addition, in place, 1237  
 mlib\_VectorMulSAdd\_U8C\_U8C\_Mod — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulSAdd\_U8C\_U8C\_Sat — vector multiplication by scalar plus addition, 1239  
 mlib\_VectorMulShift\_S16\_Mod — vector multiplication with shifting, in place, 1242  
 mlib\_VectorMulShift\_S16\_S16\_Mod — vector multiplication with shifting, 1244  
 mlib\_VectorMulShift\_S16\_S16\_Sat — vector multiplication with shifting, 1244  
 mlib\_VectorMulShift\_S16\_Sat — vector multiplication with shifting, in place, 1242  
 mlib\_VectorMulShift\_S16C\_Mod — vector multiplication with shifting, in place, 1242  
 mlib\_VectorMulShift\_S16C\_S16C\_Mod — vector multiplication with shifting, 1244



mlib\_VectorMulSShift\_S8\_Mod — vector multiplication by scalar plus shifting, in place, 1246  
 mlib\_VectorMulSShift\_S8\_S8\_Mod — vector multiplication by scalar plus shifting, 1248  
 mlib\_VectorMulSShift\_S8\_S8\_Sat — vector multiplication by scalar plus shifting, 1248  
 mlib\_VectorMulSShift\_S8\_Sat — vector multiplication by scalar plus shifting, in place, 1246  
 mlib\_VectorMulSShift\_S8C\_Mod — vector multiplication by scalar plus shifting, in place, 1246  
 mlib\_VectorMulSShift\_S8C\_S8C\_Mod — vector multiplication by scalar plus shifting, 1248  
 mlib\_VectorMulSShift\_S8C\_S8C\_Sat — vector multiplication by scalar plus shifting, 1248  
 mlib\_VectorMulSShift\_S8C\_Sat — vector multiplication by scalar plus shifting, in place, 1246  
 mlib\_VectorMulSShift\_U8\_Mod — vector multiplication by scalar plus shifting, in place, 1246  
 mlib\_VectorMulSShift\_U8\_Sat — vector multiplication by scalar plus shifting, in place, 1246  
 mlib\_VectorMulSShift\_U8\_U8\_Mod — vector multiplication by scalar plus shifting, 1248  
 mlib\_VectorMulSShift\_U8\_U8\_Sat — vector multiplication by scalar plus shifting, 1248  
 mlib\_VectorMulSShift\_U8C\_Mod — vector multiplication by scalar plus shifting, in place, 1246  
 mlib\_VectorMulSShift\_U8C\_Sat — vector multiplication by scalar plus shifting, in place, 1246  
 mlib\_VectorMulSShift\_U8C\_U8C\_Mod — vector multiplication by scalar plus shifting, 1248  
 mlib\_VectorMulSShift\_U8C\_U8C\_Sat — vector multiplication by scalar plus shifting, 1248  
 mlib\_VectorNorm\_S16\_Sat — vector norm, 1260  
 mlib\_VectorNorm\_S32\_Sat — vector norm, 1260  
 mlib\_VectorNorm\_S8\_Sat — vector norm, 1260  
 mlib\_VectorNorm\_U8\_Sat — vector norm, 1260  
 mlib\_VectorReverseByteOrder — reverse byte order of vector, 1261  
 mlib\_VectorReverseByteOrder\_D64 — reverse byte order of vector, in place, 1263  
 mlib\_VectorReverseByteOrder\_D64\_D64 — reverse byte order of vector, 1265  
 mlib\_VectorReverseByteOrder\_F32 — reverse byte order of vector, in place, 1263  
 mlib\_VectorReverseByteOrder\_F32\_F32 — reverse byte order of vector, 1265  
 mlib\_VectorReverseByteOrder\_Inp — reverse byte order of vector, in place, 1262  
 mlib\_VectorReverseByteOrder\_S16 — reverse byte order of vector, in place, 1263  
 mlib\_VectorReverseByteOrder\_S16\_S16 — reverse byte order of vector, 1265  
 mlib\_VectorReverseByteOrder\_S32 — reverse byte order of vector, in place, 1263  
 mlib\_VectorReverseByteOrder\_S32\_S32 — reverse byte order of vector, 1265  
 mlib\_VectorReverseByteOrder\_S64 — reverse byte order of vector, in place, 1263  
 mlib\_VectorReverseByteOrder\_S64\_S64 — reverse byte order of vector, 1265  
 mlib\_VectorReverseByteOrder\_U16 — reverse byte order of vector, in place, 1263  
 mlib\_VectorReverseByteOrder\_U16\_U16 — reverse byte order of vector, 1265  
 mlib\_VectorReverseByteOrder\_U32 — reverse byte order of vector, in place, 1263  
 mlib\_VectorReverseByteOrder\_U32\_U32 — reverse byte order of vector, 1265  
 mlib\_VectorReverseByteOrder\_U64 — reverse byte order of vector, in place, 1263  
 mlib\_VectorReverseByteOrder\_U64\_U64 — reverse byte order of vector, 1265  
 mlib\_VectorScale\_S16\_Mod — vector linear scaling, in place, 1267  
 mlib\_VectorScale\_S16\_S16\_Mod — vector linear scaling, 1269  
 mlib\_VectorScale\_S16\_S16\_Sat — vector linear scaling, 1269  
 mlib\_VectorScale\_S16\_S8\_Mod — vector linear scaling, 1269  
 mlib\_VectorScale\_S16\_S8\_Sat — vector linear scaling, 1269  
 mlib\_VectorScale\_S16\_Sat — vector linear scaling, in place, 1267

mlib\_VectorScale\_S16\_U8\_Mod — vector linear scaling, 1269  
 mlib\_VectorScale\_S16\_U8\_Sat — vector linear scaling, 1269  
 mlib\_VectorScale\_S16C\_Mod — vector linear scaling, in place, 1267  
 mlib\_VectorScale\_S16C\_S16C\_Mod — vector linear scaling, 1269  
 mlib\_VectorScale\_S16C\_S16C\_Sat — vector linear scaling, 1269  
 mlib\_VectorScale\_S16C\_S8C\_Mod — vector linear scaling, 1269  
 mlib\_VectorScale\_S16C\_S8C\_Sat — vector linear scaling, 1269  
 mlib\_VectorScale\_S16C\_Sat — vector linear scaling, in place, 1267  
 mlib\_VectorScale\_S16C\_U8C\_Mod — vector linear scaling, 1269  
 mlib\_VectorScale\_S16C\_U8C\_Sat — vector linear scaling, 1269  
 mlib\_VectorScale\_S32\_Mod — vector linear scaling, in place, 1267  
 mlib\_VectorScale\_S32\_S16\_Mod — vector linear scaling, 1269  
 mlib\_VectorScale\_S32\_S16\_Sat — vector linear scaling, 1269  
 mlib\_VectorScale\_S32\_S32\_Mod — vector linear scaling, 1269  
 mlib\_VectorScale\_S32\_S32\_Sat — vector linear scaling, 1269  
 mlib\_VectorScale\_S32\_Sat — vector linear scaling, in place, 1267  
 mlib\_VectorScale\_S32C\_Mod — vector linear scaling, in place, 1267  
 mlib\_VectorScale\_S32C\_S16C\_Mod — vector linear scaling, 1269  
 mlib\_VectorScale\_S32C\_S16C\_Sat — vector linear scaling, 1269  
 mlib\_VectorScale\_S32C\_S32C\_Mod — vector linear scaling, 1269  
 mlib\_VectorScale\_S32C\_S32C\_Sat — vector linear scaling, 1269  
 mlib\_VectorScale\_S32C\_Sat — vector linear scaling, in place, 1267  
 mlib\_VectorScale\_S8\_Mod — vector linear scaling, in place, 1267  
 mlib\_VectorScale\_S8\_S8\_Mod — vector linear scaling, 1269  
 mlib\_VectorScale\_S8\_S8\_Sat — vector linear scaling, 1269  
 mlib\_VectorScale\_S8\_Sat — vector linear scaling, in place, 1267  
 mlib\_VectorScale\_S8C\_Mod — vector linear scaling, in place, 1267  
 mlib\_VectorScale\_S8C\_S8C\_Mod — vector linear scaling, 1269  
 mlib\_VectorScale\_S8C\_S8C\_Sat — vector linear scaling, 1269  
 mlib\_VectorScale\_S8C\_Sat — vector linear scaling, in place, 1267  
 mlib\_VectorScale\_U8\_Mod — vector linear scaling, in place, 1267  
 mlib\_VectorScale\_U8\_Sat — vector linear scaling, in place, 1267  
 mlib\_VectorScale\_U8\_U8\_Mod — vector linear scaling, 1269  
 mlib\_VectorScale\_U8\_U8\_Sat — vector linear scaling, 1269  
 mlib\_VectorScale\_U8C\_Mod — vector linear scaling, in place, 1267  
 mlib\_VectorScale\_U8C\_Sat — vector linear scaling, in place, 1267  
 mlib\_VectorScale\_U8C\_U8C\_Mod — vector linear scaling, 1269  
 mlib\_VectorScale\_U8C\_U8C\_Sat — vector linear scaling, 1269  
 mlib\_VectorSet\_S16 — set vector to specified value, 1272  
 mlib\_VectorSet\_S16C — set vector to specified value, 1272  
 mlib\_VectorSet\_S32 — set vector to specified value, 1272  
 mlib\_VectorSet\_S32C — set vector to specified value, 1272  
 mlib\_VectorSet\_S8 — set vector to specified value, 1272  
 mlib\_VectorSet\_S8C — set vector to specified value, 1272  
 mlib\_VectorSet\_U8 — set vector to specified value, 1272  
 mlib\_VectorSet\_U8C — set vector to specified value, 1272  
 mlib\_VectorSplit\_S16\_S16C — vector split, 1274  
 mlib\_VectorSplit\_S32\_S32C — vector split, 1274

mlib\_VectorSplit\_S8\_S8C — vector split, 1274  
 mlib\_VectorSplit\_U8\_U8C — vector split, 1274  
 mlib\_VectorSub\_S16\_Mod — vector subtraction, in place, 1280  
 mlib\_VectorSub\_S16\_S16\_Mod — vector subtraction, 1282  
 mlib\_VectorSub\_S16\_S16\_Sat — vector subtraction, 1282  
 mlib\_VectorSub\_S16\_S8\_Mod — vector subtraction, 1282  
 mlib\_VectorSub\_S16\_S8\_Sat — vector subtraction, 1282  
 mlib\_VectorSub\_S16\_Sat — vector subtraction, in place, 1280  
 mlib\_VectorSub\_S16\_U8\_Mod — vector subtraction, 1282  
 mlib\_VectorSub\_S16\_U8\_Sat — vector subtraction, 1282  
 mlib\_VectorSub\_S16C\_Mod — vector subtraction, in place, 1280  
 mlib\_VectorSub\_S16C\_S16C\_Mod — vector subtraction, 1282  
 mlib\_VectorSub\_S16C\_S16C\_Sat — vector subtraction, 1282  
 mlib\_VectorSub\_S16C\_S8C\_Mod — vector subtraction, 1282  
 mlib\_VectorSub\_S16C\_S8C\_Sat — vector subtraction, 1282  
 mlib\_VectorSub\_S16C\_Sat — vector subtraction, in place, 1280  
 mlib\_VectorSub\_S16C\_U8C\_Mod — vector subtraction, 1282  
 mlib\_VectorSub\_S16C\_U8C\_Sat — vector subtraction, 1282  
 mlib\_VectorSub\_S32\_Mod — vector subtraction, in place, 1280  
 mlib\_VectorSub\_S32\_S16\_Mod — vector subtraction, 1282  
 mlib\_VectorSub\_S32\_S16\_Sat — vector subtraction, 1282  
 mlib\_VectorSub\_S32\_S32\_Mod — vector subtraction, 1282  
 mlib\_VectorSub\_S32\_S32\_Sat — vector subtraction, 1282  
 mlib\_VectorSub\_S32\_Sat — vector subtraction, in place, 1280  
 mlib\_VectorSub\_S32C\_Mod — vector subtraction, in place, 1280  
 mlib\_VectorSub\_S32C\_S16C\_Mod — vector subtraction, 1282  
 mlib\_VectorSub\_S32C\_S16C\_Sat — vector subtraction, 1282  
 mlib\_VectorSub\_S32C\_S32C\_Mod — vector subtraction, 1282  
 mlib\_VectorSub\_S32C\_S32C\_Sat — vector subtraction, 1282  
 mlib\_VectorSub\_S32C\_Sat — vector subtraction, in place, 1280  
 mlib\_VectorSub\_S8\_Mod — vector subtraction, in place, 1280  
 mlib\_VectorSub\_S8\_S8\_Mod — vector subtraction, 1282  
 mlib\_VectorSub\_S8\_S8\_Sat — vector subtraction, 1282  
 mlib\_VectorSub\_S8\_Sat — vector subtraction, in place, 1280  
 mlib\_VectorSub\_S8C\_Mod — vector subtraction, in place, 1280  
 mlib\_VectorSub\_S8C\_S8C\_Mod — vector subtraction, 1282  
 mlib\_VectorSub\_S8C\_S8C\_Sat — vector subtraction, 1282  
 mlib\_VectorSub\_S8C\_Sat — vector subtraction, in place, 1280  
 mlib\_VectorSub\_U8\_Mod — vector subtraction, in place, 1280  
 mlib\_VectorSub\_U8\_Sat — vector subtraction, in place, 1280  
 mlib\_VectorSub\_U8\_U8\_Mod — vector subtraction, 1282  
 mlib\_VectorSub\_U8\_U8\_Sat — vector subtraction, 1282  
 mlib\_VectorSub\_U8C\_Mod — vector subtraction, in place, 1280  
 mlib\_VectorSub\_U8C\_Sat — vector subtraction, in place, 1280  
 mlib\_VectorSub\_U8C\_U8C\_Mod — vector subtraction, 1282  
 mlib\_VectorSub\_U8C\_U8C\_Sat — vector subtraction, 1282  
 mlib\_VectorSubS\_S16\_Mod — vector subtraction from scalar, in place, 1275  
 mlib\_VectorSubS\_S16\_S16\_Mod — vector subtraction from scalar, 1277  
 mlib\_VectorSubS\_S16\_S16\_Sat — vector subtraction from scalar, 1277



mlib\_VectorSubS\_S16\_S8\_Mod — vector subtraction from scalar, 1277  
 mlib\_VectorSubS\_S16\_S8\_Sat — vector subtraction from scalar, 1277  
 mlib\_VectorSubS\_S16\_Sat — vector subtraction from scalar, in place, 1275  
 mlib\_VectorSubS\_S16\_U8\_Mod — vector subtraction from scalar, 1277  
 mlib\_VectorSubS\_S16\_U8\_Sat — vector subtraction from scalar, 1277  
 mlib\_VectorSubS\_S16C\_Mod — vector subtraction from scalar, in place, 1275  
 mlib\_VectorSubS\_S16C\_S16C\_Mod — vector subtraction from scalar, 1277  
 mlib\_VectorSubS\_S16C\_S16C\_Sat — vector subtraction from scalar, 1277  
 mlib\_VectorSubS\_S16C\_S8C\_Mod — vector subtraction from scalar, 1277  
 mlib\_VectorSubS\_S16C\_S8C\_Sat — vector subtraction from scalar, 1277  
 mlib\_VectorSubS\_S16C\_Sat — vector subtraction from scalar, in place, 1275  
 mlib\_VectorSubS\_S16C\_U8C\_Mod — vector subtraction from scalar, 1277  
 mlib\_VectorSubS\_S16C\_U8C\_Sat — vector subtraction from scalar, 1277  
 mlib\_VectorSubS\_S32\_Mod — vector subtraction from scalar, in place, 1275  
 mlib\_VectorSubS\_S32\_S16\_Mod — vector subtraction from scalar, 1277  
 mlib\_VectorSubS\_S32\_S16\_Sat — vector subtraction from scalar, 1277  
 mlib\_VectorSubS\_S32\_S32\_Mod — vector subtraction from scalar, 1277  
 mlib\_VectorSubS\_S32\_S32\_Sat — vector subtraction from scalar, 1277  
 mlib\_VectorSubS\_S32\_Sat — vector subtraction from scalar, in place, 1275  
 mlib\_VectorSubS\_S32C\_Mod — vector subtraction from scalar, in place, 1275  
 mlib\_VectorSubS\_S32C\_S16C\_Mod — vector subtraction from scalar, 1277  
 mlib\_VectorSubS\_S32C\_S16C\_Sat — vector subtraction from scalar, 1277  
 mlib\_VectorSubS\_S32C\_S32C\_Mod — vector subtraction from scalar, 1277  
 mlib\_VectorSubS\_S32C\_S32C\_Sat — vector subtraction from scalar, 1277  
 mlib\_VectorSubS\_S32C\_Sat — vector subtraction from scalar, in place, 1275  
 mlib\_VectorSubS\_S8\_Mod — vector subtraction from scalar, in place, 1275  
 mlib\_VectorSubS\_S8\_S8\_Mod — vector subtraction from scalar, 1277  
 mlib\_VectorSubS\_S8\_S8\_Sat — vector subtraction from scalar, 1277  
 mlib\_VectorSubS\_S8\_Sat — vector subtraction from scalar, in place, 1275  
 mlib\_VectorSubS\_S8C\_Mod — vector subtraction from scalar, in place, 1275  
 mlib\_VectorSubS\_S8C\_S8C\_Mod — vector subtraction from scalar, 1277  
 mlib\_VectorSubS\_S8C\_S8C\_Sat — vector subtraction from scalar, 1277  
 mlib\_VectorSubS\_S8C\_Sat — vector subtraction from scalar, in place, 1275  
 mlib\_VectorSubS\_U8\_Mod — vector subtraction from scalar, in place, 1275  
 mlib\_VectorSubS\_U8\_Sat — vector subtraction from scalar, in place, 1275  
 mlib\_VectorSubS\_U8\_U8\_Mod — vector subtraction from scalar, 1277  
 mlib\_VectorSubS\_U8\_U8\_Sat — vector subtraction from scalar, 1277  
 mlib\_VectorSubS\_U8C\_Mod — vector subtraction from scalar, in place, 1275  
 mlib\_VectorSubS\_U8C\_Sat — vector subtraction from scalar, in place, 1275  
 mlib\_VectorSubS\_U8C\_U8C\_Mod — vector subtraction from scalar, 1277  
 mlib\_VectorSubS\_U8C\_U8C\_Sat — vector subtraction from scalar, 1277  
 mlib\_VectorSumAbs\_S16\_Sat — sum of the absolute values of a vector, 1286  
 mlib\_VectorSumAbs\_S32\_Sat — sum of the absolute values of a vector, 1286  
 mlib\_VectorSumAbs\_S8\_Sat — sum of the absolute values of a vector, 1286  
 mlib\_VectorSumAbs\_U8\_Sat — sum of the absolute values of a vector, 1286  
 mlib\_VectorSumAbsDiff\_S16\_Sat — sum of the absolute values of the differences of two vectors, 1285  
 mlib\_VectorSumAbsDiff\_S32\_Sat — sum of the absolute values of the differences of two vectors, 1285

**mllib\_VectorSumAbsDiff\_S8\_Sat** — sum of the absolute values of the differences of two vectors, 1285  
**mllib\_VectorSumAbsDiff\_U8\_Sat** — sum of the absolute values of the differences of two vectors, 1285  
**mllib\_VectorZero\_S16** — initialize vector to zero, 1287  
**mllib\_VectorZero\_S16C** — initialize vector to zero, 1287  
**mllib\_VectorZero\_S32** — initialize vector to zero, 1287  
**mllib\_VectorZero\_S32C** — initialize vector to zero, 1287  
**mllib\_VectorZero\_S8** — initialize vector to zero, 1287  
**mllib\_VectorZero\_S8C** — initialize vector to zero, 1287  
**mllib\_VectorZero\_U8** — initialize vector to zero, 1287  
**mllib\_VectorZero\_U8C** — initialize vector to zero, 1287  
**mllib\_version** — return a version string, 1288  
**mllib\_VideoAddBlock\_U8\_S16** — adds motion-compensated 8x8 block to the current block, 1289  
**mllib\_VideoColorABGR2JFIFYCC420** — ABGR to JFIF YCbCr color conversion, 1291  
**mllib\_VideoColorABGR2JFIFYCC422** — ABGR to JFIF YCbCr color conversion, 1292  
**mllib\_VideoColorABGR2JFIFYCC444** — ABGR to JFIF YCbCr color conversion, 1293  
**mllib\_VideoColorABGR2RGB** — color conversion, 1294  
**mllib\_VideoColorABGRint\_to\_ARGBint** — convert ABGR interleaved to ARGB, 1295  
**mllib\_VideoColorARGB2JFIFYCC420** — ARGB to JFIF YCbCr color conversion, 1296  
**mllib\_VideoColorARGB2JFIFYCC422** — ARGB to JFIF YCbCr color conversion, 1297  
**mllib\_VideoColorARGB2JFIFYCC444** — ARGB to JFIF YCbCr color conversion, 1298  
**mllib\_VideoColorARGB2RGB** — color conversion, 1299  
**mllib\_VideoColorBGRaint\_to\_ABGRint** — convert BGRA interleaved to ABGR, 1300  
**mllib\_VideoColorBGRint\_to\_ABGRint** — convert BGR interleaved to ABGR interleaved, 1301  
**mllib\_VideoColorBlendABGR** — image blend, 1303  
**mllib\_VideoColorBlendABGR\_Inp** — in-place image blend, 1305  
**mllib\_VideoColorBlendABGR\_ResetAlpha** — image blend, 1303  
**mllib\_VideoColorBlendABGR\_ResetAlpha\_Inp** — in-place image blend, 1305  
**mllib\_VideoColorCMYK2JFIFYCCK444** — CMYK to JFIF YCbCr color conversion, 1307  
**mllib\_VideoColorJFIFYCC2ABGR444** — JFIF YCbCr to ABGR color conversion, 1308  
**mllib\_VideoColorJFIFYCC2ARGB444** — JFIF YCbCr to ARGB color conversion, 1309  
**mllib\_VideoColorJFIFYCC2RGB420** — JFIF YCbCr to RGB color conversion, 1310  
**mllib\_VideoColorJFIFYCC2RGB420\_Nearest** — JFIF YCbCr to RGB color conversion, 1312  
**mllib\_VideoColorJFIFYCC2RGB422** — JFIF YCbCr to RGB color conversion, 1313  
**mllib\_VideoColorJFIFYCC2RGB422\_Nearest** — JFIF YCbCr to RGB color conversion, 1314  
**mllib\_VideoColorJFIFYCC2RGB444** — JFIF YCbCr to RGB color conversion, 1315  
**mllib\_VideoColorJFIFYCC2RGB444\_S16** — JFIF YCbCr to RGB color conversion, 1316  
**mllib\_VideoColorJFIFYCCK2CMYK444** — JFIF YCbCr to CMYK color conversion, 1317  
**mllib\_VideoColorMerge2** — color conversion (color channel merge), 1318  
**mllib\_VideoColorMerge2\_S16** — color conversion (color channel merge), 1319  
**mllib\_VideoColorMerge3** — color conversion (color channel merge), 1320  
**mllib\_VideoColorMerge3\_S16** — color conversion (color channel merge), 1321  
**mllib\_VideoColorMerge4** — color conversion (color channel merge), 1322  
**mllib\_VideoColorMerge4\_S16** — color conversion (color channel merge), 1323  
**mllib\_VideoColorResizeABGR** — image resize, 1324  
**mllib\_VideoColorRGB2ABGR** — color conversion, 1325

mlib\_VideoColorRGB2ARGB — color conversion, 1326  
 mlib\_VideoColorRGB2JFIFYCC420 — RGB to JFIF YCbCr color conversion, 1327  
 mlib\_VideoColorRGB2JFIFYCC422 — RGB to JFIF YCbCr color conversion, 1328  
 mlib\_VideoColorRGB2JFIFYCC444 — RGB to JFIF YCbCr color conversion, 1329  
 mlib\_VideoColorRGB2JFIFYCC444\_S16 — RGB to JFIF YCbCr color conversion, 1330  
 mlib\_VideoColorRGBAint\_to\_ABGRint — convert RGBA interleaved to ABGR, 1331  
 mlib\_VideoColorRGBint\_to\_ABGRint — convert RGB interleaved to ABGR interleaved, 1332  
 mlib\_VideoColorRGBseq\_to\_ABGRint — convert RGB sequential to ABGR interleaved, 1334  
 mlib\_VideoColorRGBXint\_to\_ABGRint — convert RGBX interleaved to ABGR interleaved, 1336  
 mlib\_VideoColorRGBXint\_to\_ARGBint — convert RGBX interleaved to ARGB interleaved, 1338  
 mlib\_VideoColorSplit2 — color conversion (color channel split), 1339  
 mlib\_VideoColorSplit2\_S16 — color conversion (color channel split), 1340  
 mlib\_VideoColorSplit3 — color conversion (color channel split), 1341  
 mlib\_VideoColorSplit3\_S16 — color conversion (color channel split), 1342  
 mlib\_VideoColorSplit4 — color conversion (color channel split), 1343  
 mlib\_VideoColorSplit4\_S16 — color conversion (color channel split), 1344  
 mlib\_VideoColorUYV444int\_to\_ABGRint — color convert UYV interleaved to ABGR interleaved, 1345  
 mlib\_VideoColorUYV444int\_to\_ARGBint — color convert UYV interleaved to ARGB interleaved, 1347  
 mlib\_VideoColorUYV444int\_to\_UYVY422int — convert UYV interleaved with subsampling, 1349  
 mlib\_VideoColorUYV444int\_to\_YUYV422int — convert UYV interleaved with subsampling, 1350  
 mlib\_VideoColorUYVY422int\_to\_ABGRint — color convert UYVY interleaved to ABGR interleaved, 1351  
 mlib\_VideoColorUYVY422int\_to\_ARGBint — color convert UYVY interleaved to ARGB interleaved, 1353  
 mlib\_VideoColorXRGBint\_to\_ABGRint — convert XRGB interleaved to ABGR interleaved, 1355  
 mlib\_VideoColorXRGBint\_to\_ARGBint — convert XRGB interleaved to ARGB interleaved, 1356  
 mlib\_VideoColorYUV2ABGR411 — YUV to RGB color conversion, 1357  
 mlib\_VideoColorYUV2ABGR420 — YUV to RGB color conversion, 1359  
 mlib\_VideoColorYUV2ABGR420\_W — YUV to RGB color conversion, 1361  
 mlib\_VideoColorYUV2ABGR420\_WX2 — YUV to RGB color conversion, 1363  
 mlib\_VideoColorYUV2ABGR420\_WX3 — YUV to RGB color conversion, 1365  
 mlib\_VideoColorYUV2ABGR420\_X2 — YUV to RGB color conversion, 1367  
 mlib\_VideoColorYUV2ABGR420\_X3 — YUV to RGB color conversion, 1369  
 mlib\_VideoColorYUV2ABGR422 — YUV to RGB color conversion, 1371  
 mlib\_VideoColorYUV2ABGR444 — YUV to RGB color conversion, 1373  
 mlib\_VideoColorYUV2ARGB411 — YUV to RGB color conversion, 1375  
 mlib\_VideoColorYUV2ARGB420 — YUV to RGB color conversion, 1377  
 mlib\_VideoColorYUV2ARGB422 — YUV to RGB color conversion, 1379  
 mlib\_VideoColorYUV2ARGB444 — YUV to RGB color conversion, 1381  
 mlib\_VideoColorYUV2RGB411 — YUV to RGB color conversion, 1383  
 mlib\_VideoColorYUV2RGB420 — YUV to RGB color conversion, 1385  
 mlib\_VideoColorYUV2RGB422 — YUV to RGB color conversion, 1387  
 mlib\_VideoColorYUV2RGB444 — YUV to RGB color conversion, 1389

**mllib\_VideoColorYUV411seq\_to\_ABGRint** — color convert YUV sequential to ABGR interleaved, 1391  
**mllib\_VideoColorYUV411seq\_to\_ARGBint** — color convert YUV sequential to ARGB interleaved, 1393  
**mllib\_VideoColorYUV411seq\_to\_UYVY422int** — convert YUV sequential to interleaved, 1395  
**mllib\_VideoColorYUV411seq\_to\_YUYV422int** — convert YUV sequential to interleaved, 1397  
**mllib\_VideoColorYUV420seq\_to\_ABGRint** — color convert YUV sequential to ABGR interleaved, 1399  
**mllib\_VideoColorYUV420seq\_to\_ARGBint** — color convert YUV sequential to ARGB interleaved, 1401  
**mllib\_VideoColorYUV420seq\_to\_UYVY422int** — convert YUV sequential to interleaved, 1403  
**mllib\_VideoColorYUV420seq\_to\_YUYV422int** — convert YUV sequential to interleaved, 1405  
**mllib\_VideoColorYUV422seq\_to\_ABGRint** — color convert YUV sequential to ABGR interleaved, 1407  
**mllib\_VideoColorYUV422seq\_to\_ARGBint** — color convert YUV sequential to ARGB interleaved, 1409  
**mllib\_VideoColorYUV422seq\_to\_UYVY422int** — convert YUV sequential to interleaved, 1411  
**mllib\_VideoColorYUV422seq\_to\_YUYV422int** — convert YUV sequential to interleaved, 1413  
**mllib\_VideoColorYUV444int\_to\_ABGRint** — color convert YUV interleaved to ABGR interleaved, 1415  
**mllib\_VideoColorYUV444int\_to\_ARGBint** — color convert YUV interleaved to ARGB interleaved, 1417  
**mllib\_VideoColorYUV444int\_to\_UYVY422int** — convert YUV interleaved with subsampling, 1419  
**mllib\_VideoColorYUV444int\_to\_YUYV422int** — convert YUV interleaved with subsampling, 1420  
**mllib\_VideoColorYUV444seq\_to\_ABGRint** — color convert YUV sequential to ABGR interleaved, 1421  
**mllib\_VideoColorYUV444seq\_to\_ARGBint** — color convert YUV sequential to ARGB interleaved, 1423  
**mllib\_VideoColorYUV444seq\_to\_UYVY422int** — convert YUV sequential to interleaved with subsampling, 1425  
**mllib\_VideoColorYUV444seq\_to\_YUYV422int** — convert YUV sequential to interleaved with subsampling, 1426  
**mllib\_VideoColorYUYV422int\_to\_ABGRint** — color convert YUYV interleaved to ABGR interleaved, 1427  
**mllib\_VideoColorYUYV422int\_to\_ARGBint** — color convert YUYV interleaved to ARGB interleaved, 1429  
**mllib\_VideoCopyRef\_S16\_U8** — copies a block from the reference block to the current block, 1437  
**mllib\_VideoCopyRef\_S16\_U8\_16x16** — copies a block from the reference block to the current block, 1435  
**mllib\_VideoCopyRef\_S16\_U8\_16x8** — copies a block from the reference block to the current block, 1435  
**mllib\_VideoCopyRef\_S16\_U8\_8x16** — copies a block from the reference block to the current block, 1435  
**mllib\_VideoCopyRef\_S16\_U8\_8x4** — copies a block from the reference block to the current block, 1435  
**mllib\_VideoCopyRef\_S16\_U8\_8x8** — copies a block from the reference block to the current block, 1435  
**mllib\_VideoCopyRef\_U8\_U8** — copies a block from the reference block to the current block, 1441  
**mllib\_VideoCopyRef\_U8\_U8\_16x16** — copies an 8x8 block from the reference block to the current block, 1439  
**mllib\_VideoCopyRef\_U8\_U8\_16x8** — copies an 8x8 block from the reference block to the current block, 1439  
**mllib\_VideoCopyRef\_U8\_U8\_8x16** — copies an 8x8 block from the reference block to the current block, 1439  
**mllib\_VideoCopyRef\_U8\_U8\_8x4** — copies an 8x8 block from the reference block to the current block, 1439  
**mllib\_VideoCopyRef\_U8\_U8\_8x8** — copies an 8x8 block from the reference block to the current block, 1439

mlib\_VideoCopyRefAve\_U8\_U8 — copies and averages a block from the reference block to the current block, 1433  
 mlib\_VideoCopyRefAve\_U8\_U8\_16x16 — copies and averages a block from the reference block to the current block, 1431  
 mlib\_VideoCopyRefAve\_U8\_U8\_16x8 — copies and averages a block from the reference block to the current block, 1431  
 mlib\_VideoCopyRefAve\_U8\_U8\_8x16 — copies and averages a block from the reference block to the current block, 1431  
 mlib\_VideoCopyRefAve\_U8\_U8\_8x4 — copies and averages a block from the reference block to the current block, 1431  
 mlib\_VideoCopyRefAve\_U8\_U8\_8x8 — copies and averages a block from the reference block to the current block, 1431  
 mlib\_VideoDCT16x16\_S16\_S16 — forward Discrete Cosine Transform (DCT), 1443  
 mlib\_VideoDCT16x16\_S16\_S16\_B10 — forward Discrete Cosine Transform (DCT), 1444  
 mlib\_VideoDCT2x2\_S16\_S16 — forward Discrete Cosine Transform (DCT), 1445  
 mlib\_VideoDCT4x4\_S16\_S16 — forward Discrete Cosine Transform (DCT), 1446  
 mlib\_VideoDCT8x8\_S16\_S16 — forward Discrete Cosine Transform (DCT), 1447  
 mlib\_VideoDCT8x8\_S16\_S16\_B12 — forward Discrete Cosine Transform (DCT), 1448  
 mlib\_VideoDCT8x8\_S16\_S16\_NA — forward Discrete Cosine Transform (DCT), 1449  
 mlib\_VideoDCT8x8\_S16\_U8 — forward Discrete Cosine Transform (DCT), 1450  
 mlib\_VideoDCT8x8\_S16\_U8\_NA — forward Discrete Cosine Transform (DCT), 1451  
 mlib\_VideoDeQuantize\_S16 — dequantization of forward Discrete Cosine Transform (DCT) coefficients, 1453  
 mlib\_VideoDeQuantizeInit\_S16 — dequantization of forward Discrete Cosine Transform (DCT) coefficients, 1452  
 mlib\_VideoDownSample420 — down sampling rate conversion in JFIF, 1454  
 mlib\_VideoDownSample420\_S16 — down sampling rate conversion in JFIF, 1455  
 mlib\_VideoDownSample422 — down sampling rate conversion in JFIF, 1456  
 mlib\_VideoDownSample422\_S16 — down sampling rate conversion in JFIF, 1457  
 mlib\_VideoH263OverlappedMC\_S16\_U8 — generates the 8x8 luminance prediction block in the Advanced Prediction Mode for H.263 codec, 1458  
 mlib\_VideoH263OverlappedMC\_U8\_U8 — generates the 8x8 luminance prediction block in the Advanced Prediction Mode for H.263 codec, 1461  
 mlib\_VideoIDCT\_IEEE\_S16\_S16 — IEEE-1180 compliant inverse Discrete Cosine Transform, 1473  
 mlib\_VideoIDCT8x8\_S16\_S16 — inverse Discrete Cosine Transform, 1464  
 mlib\_VideoIDCT8x8\_S16\_S16\_DC — inverse Discrete Cosine Transform, 1465  
 mlib\_VideoIDCT8x8\_S16\_S16\_NA — inverse Discrete Cosine Transform, 1466  
 mlib\_VideoIDCT8x8\_S16\_S16\_Q1 — inverse Discrete Cosine Transform, 1467  
 mlib\_VideoIDCT8x8\_S16\_S16\_Q1\_Mismatch — inverse Discrete Cosine Transform, 1468  
 mlib\_VideoIDCT8x8\_U8\_S16 — inverse Discrete Cosine Transform, 1469  
 mlib\_VideoIDCT8x8\_U8\_S16\_DC — inverse Discrete Cosine Transform, 1470  
 mlib\_VideoIDCT8x8\_U8\_S16\_NA — inverse Discrete Cosine Transform, 1471  
 mlib\_VideoIDCT8x8\_U8\_S16\_Q1 — inverse Discrete Cosine Transform, 1472  
 mlib\_VideoInterpAveX\_U8\_U8 — half-pixel interpolation in the X direction and averaging for reference block, 1476  
 mlib\_VideoInterpAveX\_U8\_U8\_16x16 — half-pixel interpolation in the X direction and averaging for reference block, 1474  
 mlib\_VideoInterpAveX\_U8\_U8\_16x8 — half-pixel interpolation in the X direction and averaging for reference block, 1474  
 mlib\_VideoInterpAveX\_U8\_U8\_8x16 — half-pixel interpolation in the X direction and averaging for reference block, 1474  
 mlib\_VideoInterpAveX\_U8\_U8\_8x4 — half-pixel interpolation in the X direction and averaging for reference block, 1474

mlib\_VideoInterpAveX\_U8\_U8\_8x8 — half-pixel interpolation in the X direction and averaging for reference block, 1474  
 mlib\_VideoInterpAveXY\_U8\_U8 — half-pixel interpolation in the X and Y directions and averaging for reference block, 1480  
 mlib\_VideoInterpAveXY\_U8\_U8\_16x16 — half-pixel interpolation in the X and Y directions and averaging for reference block, 1478  
 mlib\_VideoInterpAveXY\_U8\_U8\_16x8 — half-pixel interpolation in the X and Y directions and averaging for reference block, 1478  
 mlib\_VideoInterpAveXY\_U8\_U8\_8x16 — half-pixel interpolation in the X and Y directions and averaging for reference block, 1478  
 mlib\_VideoInterpAveXY\_U8\_U8\_8x4 — half-pixel interpolation in the X and Y directions and averaging for reference block, 1478  
 mlib\_VideoInterpAveXY\_U8\_U8\_8x8 — half-pixel interpolation in the X and Y directions and averaging for reference block, 1478  
 mlib\_VideoInterpAveY\_U8\_U8 — half-pixel interpolation in the Y direction and averaging for reference block, 1484  
 mlib\_VideoInterpAveY\_U8\_U8\_16x16 — half-pixel interpolation in the Y direction and averaging for reference block, 1482  
 mlib\_VideoInterpAveY\_U8\_U8\_16x8 — half-pixel interpolation in the Y direction and averaging for reference block, 1482  
 mlib\_VideoInterpAveY\_U8\_U8\_8x16 — half-pixel interpolation in the Y direction and averaging for reference block, 1482  
 mlib\_VideoInterpAveY\_U8\_U8\_8x4 — half-pixel interpolation in the Y direction and averaging for reference block, 1482  
 mlib\_VideoInterpAveY\_U8\_U8\_8x8 — half-pixel interpolation in the Y direction and averaging for reference block, 1482  
 mlib\_VideoInterpX\_S16\_U8 — half-pixel interpolation in the X direction, 1488  
 mlib\_VideoInterpX\_S16\_U8\_16x16 — half-pixel interpolation in the X direction, 1486  
 mlib\_VideoInterpX\_S16\_U8\_16x8 — half-pixel interpolation in the X direction, 1486  
 mlib\_VideoInterpX\_S16\_U8\_8x16 — half-pixel interpolation in the X direction, 1486  
 mlib\_VideoInterpX\_S16\_U8\_8x4 — half-pixel interpolation in the X direction, 1486  
 mlib\_VideoInterpX\_S16\_U8\_8x8 — half-pixel interpolation in the X direction, 1486  
 mlib\_VideoInterpX\_U8\_U8 — half-pixel interpolation in the X direction, 1492  
 mlib\_VideoInterpX\_U8\_U8\_16x16 — half-pixel interpolation in the X direction, 1490  
 mlib\_VideoInterpX\_U8\_U8\_16x8 — half-pixel interpolation in the X direction, 1490  
 mlib\_VideoInterpX\_U8\_U8\_8x16 — half-pixel interpolation in the X direction, 1490  
 mlib\_VideoInterpX\_U8\_U8\_8x4 — half-pixel interpolation in the X direction, 1490  
 mlib\_VideoInterpX\_U8\_U8\_8x8 — half-pixel interpolation in the X direction, 1490  
 mlib\_VideoInterpX\_Y\_XY\_U8\_U8 — half-pixel interpolation in both X and Y directions for replenishment mode, 1502  
 mlib\_VideoInterpXY\_S16\_U8 — half-pixel interpolation in the X and Y directions, 1496  
 mlib\_VideoInterpXY\_S16\_U8\_16x16 — half-pixel interpolation in the X and Y directions for motion compensation, 1494  
 mlib\_VideoInterpXY\_S16\_U8\_16x8 — half-pixel interpolation in the X and Y directions for motion compensation, 1494  
 mlib\_VideoInterpXY\_S16\_U8\_8x16 — half-pixel interpolation in the X and Y directions for motion compensation, 1494  
 mlib\_VideoInterpXY\_S16\_U8\_8x4 — half-pixel interpolation in the X and Y directions for motion compensation, 1494  
 mlib\_VideoInterpXY\_S16\_U8\_8x8 — half-pixel interpolation in the X and Y directions for motion compensation, 1494  
 mlib\_VideoInterpXY\_U8\_U8 — half-pixel interpolation in the X and Y directions, 1500  
 mlib\_VideoInterpXY\_U8\_U8\_16x16 — half-pixel interpolation in the X and Y directions, 1498  
 mlib\_VideoInterpXY\_U8\_U8\_16x8 — half-pixel interpolation in the X and Y directions, 1498

mlib\_VideoInterpXY\_U8\_U8\_8x16 — half-pixel interpolation in the X and Y directions, 1498  
 mlib\_VideoInterpXY\_U8\_U8\_8x4 — half-pixel interpolation in the X and Y directions, 1498  
 mlib\_VideoInterpXY\_U8\_U8\_8x8 — half-pixel interpolation in the X and Y directions, 1498  
 mlib\_VideoInterpY\_S16\_U8 — half-pixel interpolation in the Y direction, 1505  
 mlib\_VideoInterpY\_S16\_U8\_16x16 — half-pixel interpolation in the Y direction, 1503  
 mlib\_VideoInterpY\_S16\_U8\_16x8 — half-pixel interpolation in the Y direction, 1503  
 mlib\_VideoInterpY\_S16\_U8\_8x16 — half-pixel interpolation in the Y direction, 1503  
 mlib\_VideoInterpY\_S16\_U8\_8x4 — half-pixel interpolation in the Y direction, 1503  
 mlib\_VideoInterpY\_S16\_U8\_8x8 — half-pixel interpolation in the Y direction, 1503  
 mlib\_VideoInterpY\_U8\_U8 — half-pixel interpolation in the Y direction, 1509  
 mlib\_VideoInterpY\_U8\_U8\_16x16 — half-pixel interpolation in the Y direction, 1507  
 mlib\_VideoInterpY\_U8\_U8\_16x8 — half-pixel interpolation in the Y direction, 1507  
 mlib\_VideoInterpY\_U8\_U8\_8x16 — half-pixel interpolation in the Y direction, 1507  
 mlib\_VideoInterpY\_U8\_U8\_8x4 — half-pixel interpolation in the Y direction, 1507  
 mlib\_VideoInterpY\_U8\_U8\_8x8 — half-pixel interpolation in the Y direction, 1507  
 mlib\_VideoP64Decimate\_U8\_U8 — averages the source raster image over 2x2 blocks and writes the results to the destination raster image, 1511  
 mlib\_VideoP64Loop\_S16\_U8 — applies a 2-dimensional (2D) 3x3 spatial filter on the reference block, 1513  
 mlib\_VideoP64Loop\_U8\_U8 — applies a 2-dimensional (2D) 3x3 spatial filter on the reference block, 1515  
 mlib\_VideoQuantize\_S16 — quantization of forward Discrete Cosine Transform (DCT) coefficients, 1518  
 mlib\_VideoQuantizeInit\_S16 — quantization of forward Discrete Cosine Transform (DCT) coefficients, 1517  
 mlib\_VideoReversibleColorRGB2YUV\_S16\_S16 — reversible color space conversion for wavelet transformation, 1519  
 mlib\_VideoReversibleColorRGB2YUV\_S16\_U8 — reversible color space conversion for wavelet transformation, 1519  
 mlib\_VideoReversibleColorRGB2YUV\_S32\_S16 — reversible color space conversion for wavelet transformation, 1519  
 mlib\_VideoReversibleColorRGB2YUV\_U8\_U8 — reversible color space conversion for wavelet transformation, 1519  
 mlib\_VideoReversibleColorYUV2RGB\_S16\_S16 — reversible color space conversion for wavelet transformation, 1521  
 mlib\_VideoReversibleColorYUV2RGB\_S16\_S32 — reversible color space conversion for wavelet transformation, 1521  
 mlib\_VideoReversibleColorYUV2RGB\_U8\_S16 — reversible color space conversion for wavelet transformation, 1521  
 mlib\_VideoReversibleColorYUV2RGB\_U8\_U8 — reversible color space conversion for wavelet transformation, 1521  
 mlib\_VideoSignMagnitudeConvert\_S16 — wavelet transformation, sign-magnitude conversion, 1523  
 mlib\_VideoSignMagnitudeConvert\_S16\_S16 — wavelet transformation, sign-magnitude conversion, 1524  
 mlib\_VideoSignMagnitudeConvert\_S32 — wavelet transformation, sign-magnitude conversion, 1525  
 mlib\_VideoSignMagnitudeConvert\_S32\_S32 — wavelet transformation, sign-magnitude conversion, 1526  
 mlib\_VideoSumAbsDiff — motion estimation, 1527  
 mlib\_VideoUpSample420 — up sampling rate conversion in JFIF, 1528  
 mlib\_VideoUpSample420\_Nearest — up sampling rate conversion in JFIF, 1529  
 mlib\_VideoUpSample420\_Nearest\_S16 — up sampling rate conversion in JFIF, 1530  
 mlib\_VideoUpSample420\_S16 — up sampling rate conversion in JFIF, 1531  
 mlib\_VideoUpSample422 — up sampling rate conversion in JFIF, 1532

mlib\_VideoUpSample422\_Nearest — up sampling rate conversion in JFIF, 1533  
 mlib\_VideoUpSample422\_Nearest\_S16 — up sampling rate conversion in JFIF, 1534  
 mlib\_VideoUpSample422\_S16 — up sampling rate conversion in JFIF, 1535  
 mlib\_VideoWaveletForwardTwoTenTrans — wavelet transformation, 1536  
 mlib\_VideoWaveletForwardTwoTenTrans\_S16\_S16 — wavelet transformation, 1536  
 mlib\_VideoWaveletForwardTwoTenTrans\_S16\_U8 — wavelet transformation, 1536  
 mlib\_VideoWaveletForwardTwoTenTrans\_S32\_S16 — wavelet transformation, 1536  
 mlib\_VideoWaveletForwardTwoTenTrans\_S32\_S32 — wavelet transformation, 1536  
 mlib\_VideoWaveletInverseTwoTenTrans — wavelet transformation, 1537  
 mlib\_VideoWaveletInverseTwoTenTrans\_S16\_S16 — wavelet transformation, 1537  
 mlib\_VideoWaveletInverseTwoTenTrans\_S16\_S32 — wavelet transformation, 1537  
 mlib\_VideoWaveletInverseTwoTenTrans\_S32\_S32 — wavelet transformation, 1537  
 mlib\_VideoWaveletInverseTwoTenTrans\_U8\_S16 — wavelet transformation, 1537  
 mlib\_VolumeFindMax\_S16 — maximum intensity searching, 1540  
 mlib\_VolumeFindMax\_U8 — maximum intensity searching, 1540  
 mlib\_VolumeFindMaxBMask\_S16 — maximum intensity searching, 1538  
 mlib\_VolumeFindMaxBMask\_U8 — maximum intensity searching, 1538  
 mlib\_VolumeFindMaxCMask\_S16 — maximum intensity searching, 1539  
 mlib\_VolumeFindMaxCMask\_U8 — maximum intensity searching, 1539  
 mlib\_VolumeRayCast\_Blocked — cast a ray (or rays) through a 3D data set, 1541  
 mlib\_VolumeRayCast\_Blocked\_Divergent\_Nearest\_S16\_S16 — cast a ray (or rays) through a 3D data set, 1541  
 mlib\_VolumeRayCast\_Blocked\_Divergent\_Nearest\_U8\_U8 — cast a ray (or rays) through a 3D data set, 1541  
 mlib\_VolumeRayCast\_Blocked\_Divergent\_Trilinear\_S16\_S16 — cast a ray (or rays) through a 3D data set, 1541  
 mlib\_VolumeRayCast\_Blocked\_Divergent\_Trilinear\_U8\_U8 — cast a ray (or rays) through a 3D data set, 1541  
 mlib\_VolumeRayCast\_Blocked\_Parallel\_Nearest\_S16\_S16 — cast a ray (or rays) through a 3D data set, 1541  
 mlib\_VolumeRayCast\_Blocked\_Parallel\_Nearest\_U8\_U8 — cast a ray (or rays) through a 3D data set, 1541  
 mlib\_VolumeRayCast\_Blocked\_Parallel\_Trilinear\_S16\_S16 — cast a ray (or rays) through a 3D data set, 1541  
 mlib\_VolumeRayCast\_Blocked\_Parallel\_Trilinear\_U8\_U8 — cast a ray (or rays) through a 3D data set, 1541  
 mlib\_VolumeRayCast\_General — cast a ray (or rays) through a 3D data set, 1543  
 mlib\_VolumeRayCast\_General\_Divergent\_Nearest\_S16\_S16 — cast a ray (or rays) through a 3D data set, 1543  
 mlib\_VolumeRayCast\_General\_Divergent\_Nearest\_U8\_Bit — cast a ray (or rays) through a 3D data set, 1543  
 mlib\_VolumeRayCast\_General\_Divergent\_Nearest\_U8\_U8 — cast a ray (or rays) through a 3D data set, 1543  
 mlib\_VolumeRayCast\_General\_Divergent\_Trilinear\_S16\_S16 — cast a ray (or rays) through a 3D data set, 1543  
 mlib\_VolumeRayCast\_General\_Divergent\_Trilinear\_U8\_U8 — cast a ray (or rays) through a 3D data set, 1543  
 mlib\_VolumeRayCast\_General\_Parallel\_Nearest\_S16\_S16 — cast a ray (or rays) through a 3D data set, 1543  
 mlib\_VolumeRayCast\_General\_Parallel\_Nearest\_U8\_Bit — cast a ray (or rays) through a 3D data set, 1543  
 mlib\_VolumeRayCast\_General\_Parallel\_Nearest\_U8\_U8 — cast a ray (or rays) through a 3D data set, 1543  
 mlib\_VolumeRayCast\_General\_Parallel\_Trilinear\_S16\_S16 — cast a ray (or rays) through a 3D data set, 1543



mlib\_VolumeRayCast\_General\_Parallel\_Trilinear\_U8\_U8 — cast a ray (or rays) through a 3D data set, 1543  
 mlib\_VolumeWindowLevel — window-level operation, 1545  
 motion estimation —  
     mlib\_VideoSumAbsDiff, 1527  
 multiplication — mlib\_ImageMulShift, 605  
 multiplication — mlib\_SignalMul\_F32, 1118  
 multiplication — mlib\_SignalMul\_F32S, 1118  
 multiplication —  
     mlib\_SignalMul\_F32S\_F32S, 1119  
 multiplication —  
     mlib\_SignalMul\_F32\_F32, 1119  
 multiplication —  
     mlib\_SignalMul\_S16S\_S16S\_Sat, 1136  
 multiplication —  
     mlib\_SignalMul\_S16S\_Sat, 1137  
 multiplication —  
     mlib\_SignalMul\_S16\_S16\_Sat, 1136  
 multiplication —  
     mlib\_SignalMul\_S16\_Sat, 1137  
 multiplication by a scalar —  
     mlib\_SignalMulS\_F32, 1142  
 multiplication by a scalar —  
     mlib\_SignalMulS\_F32S, 1142  
 multiplication by a scalar —  
     mlib\_SignalMulS\_F32S\_F32S, 1143  
 multiplication by a scalar —  
     mlib\_SignalMulS\_F32\_F32, 1143  
 multiplication by a scalar —  
     mlib\_SignalMulS\_S16S\_S16S\_Sat, 1146  
 multiplication by a scalar —  
     mlib\_SignalMulS\_S16S\_Sat, 1147  
 multiplication by a scalar —  
     mlib\_SignalMulS\_S16\_S16\_Sat, 1146  
 multiplication by a scalar —  
     mlib\_SignalMulS\_S16\_Sat, 1147  
 multiplication by a scalar plus addition —  
     mlib\_SignalMulSAdd\_F32, 1138  
 multiplication by a scalar plus addition —  
     mlib\_SignalMulSAdd\_F32S, 1138  
 multiplication by a scalar plus addition —  
     mlib\_SignalMulSAdd\_F32S\_F32S, 1139  
 multiplication by a scalar plus addition —  
     mlib\_SignalMulSAdd\_F32\_F32, 1139  
 multiplication by a scalar plus addition —  
     mlib\_SignalMulSAdd\_S16S\_S16S\_Sat, 1140  
 multiplication by a scalar plus addition —  
     mlib\_SignalMulSAdd\_S16S\_Sat, 1141  
 multiplication by a scalar plus addition —  
     mlib\_SignalMulSAdd\_S16\_S16\_Sat, 1140  
 multiplication by a scalar plus addition —  
     mlib\_SignalMulSAdd\_S16\_Sat, 1141  
 multiplication by a scalar plus addition —  
     mlib\_SignalMulSShiftAdd\_S16S\_S16S\_Sat, 1148  
 multiplication by a scalar plus addition —  
     mlib\_SignalMulSShiftAdd\_S16S\_Sat, 1149  
 multiplication by a scalar plus addition —  
     mlib\_SignalMulSShiftAdd\_S16\_S16\_Sat, 1148  
 multiplication by a scalar plus addition —  
     mlib\_SignalMulSShiftAdd\_S16\_Sat, 1149  
 multiplication by a scalar with shifting —  
     mlib\_SignalMulSShift\_S16S\_S16S\_Sat, 1150  
 multiplication by a scalar with shifting —  
     mlib\_SignalMulSShift\_S16S\_Sat, 1151  
 multiplication by a scalar with shifting —  
     mlib\_SignalMulSShift\_S16\_S16\_Sat, 1150  
 multiplication by a scalar with shifting —  
     mlib\_SignalMulSShift\_S16\_Sat, 1151  
 multiplication of vector by matrix —  
     mlib\_VectorMulM\_S16C\_S16C\_Mod, 1234  
 multiplication of vector by matrix —  
     mlib\_VectorMulM\_S16C\_S16C\_Sat, 1234  
 multiplication of vector by matrix —  
     mlib\_VectorMulM\_S16C\_S8C\_Mod, 1234  
 multiplication of vector by matrix —  
     mlib\_VectorMulM\_S16C\_S8C\_Sat, 1234  
 multiplication of vector by matrix —  
     mlib\_VectorMulM\_S16C\_U8C\_Mod, 1234  
 multiplication of vector by matrix —  
     mlib\_VectorMulM\_S16C\_U8C\_Sat, 1234  
 multiplication of vector by matrix —  
     mlib\_VectorMulM\_S16\_S16\_Mod, 1234  
 multiplication of vector by matrix —  
     mlib\_VectorMulM\_S16\_S16\_Sat, 1234  
 multiplication of vector by matrix —  
     mlib\_VectorMulM\_S16\_S8\_Mod, 1234  
 multiplication of vector by matrix —  
     mlib\_VectorMulM\_S16\_S8\_Sat, 1234  
 multiplication of vector by matrix —  
     mlib\_VectorMulM\_S16\_U8\_Mod, 1234  
 multiplication of vector by matrix —  
     mlib\_VectorMulM\_S16\_U8\_Sat, 1234  
 multiplication of vector by matrix —  
     mlib\_VectorMulM\_S32C\_S16C\_Mod, 1234

multiplication of vector by matrix —  
  mllib\_VectorMulM\_S32C\_S16C\_Sat, 1234  
multiplication of vector by matrix —  
  mllib\_VectorMulM\_S32C\_S32C\_Mod, 1234  
multiplication of vector by matrix —  
  mllib\_VectorMulM\_S32C\_S32C\_Sat, 1234  
multiplication of vector by matrix —  
  mllib\_VectorMulM\_S32\_S16\_Mod, 1234  
multiplication of vector by matrix —  
  mllib\_VectorMulM\_S32\_S16\_Sat, 1234  
multiplication of vector by matrix —  
  mllib\_VectorMulM\_S32\_S32\_Mod, 1234  
multiplication of vector by matrix —  
  mllib\_VectorMulM\_S32\_S32\_Sat, 1234  
multiplication of vector by matrix —  
  mllib\_VectorMulM\_S8C\_S8C\_Mod, 1234  
multiplication of vector by matrix —  
  mllib\_VectorMulM\_S8C\_S8C\_Sat, 1234  
multiplication of vector by matrix —  
  mllib\_VectorMulM\_S8\_S8\_Mod, 1234  
multiplication of vector by matrix —  
  mllib\_VectorMulM\_S8\_S8\_Sat, 1234  
multiplication of vector by matrix —  
  mllib\_VectorMulM\_U8C\_U8C\_Mod, 1234  
multiplication of vector by matrix —  
  mllib\_VectorMulM\_U8C\_U8C\_Sat, 1234  
multiplication of vector by matrix —  
  mllib\_VectorMulM\_U8\_U8\_Mod, 1234  
multiplication of vector by matrix —  
  mllib\_VectorMulM\_U8\_U8\_Sat, 1234  
multiplication of vector by matrix with shifting  
—  
  mllib\_VectorMulMShift\_S16C\_S16C\_Mod, 1232  
multiplication of vector by matrix with shifting  
—  
  mllib\_VectorMulMShift\_S16C\_S16C\_Sat, 1232  
multiplication of vector by matrix with shifting  
—  
  mllib\_VectorMulMShift\_S16\_S16\_Mod, 1232  
multiplication of vector by matrix with shifting  
— mllib\_VectorMulMShift\_S16\_S16\_Sat, 1232  
multiplication with shifting —  
  mllib\_SignalMulShift\_S16S\_S16S\_Sat, 1144  
multiplication with shifting —  
  mllib\_SignalMulShift\_S16S\_Sat, 1145  
multiplication with shifting —  
  mllib\_SignalMulShift\_S16\_S16\_Sat, 1144  
multiplication with shifting —  
  mllib\_SignalMulShift\_S16\_Sat, 1145  
multiplication, in place —  
  mllib\_ImageMulShift\_Inp, 606  
multiplication with a constant —  
  mllib\_ImageConstMul, 349  
multiply with a constant —  
  mllib\_ImageConstMul\_Fp, 350  
multiply with a constant —  
  mllib\_ImageConstMul\_Fp\_Inp, 351  
multiply with a constant —  
  mllib\_ImageConstMul\_Inp, 352  
multiply with a constant, with shifting —  
  mllib\_ImageConstMulShift, 353  
multiply with a constant, with shifting —  
  mllib\_ImageConstMulShift\_Inp, 354  
MxN convolution —  
  mllib\_ImageConvMxN, 403  
MxN convolution —  
  mllib\_ImageConvMxN\_Fp, 405  
MxN convolution, with kernel analysis for  
taking advantage of special cases —  
  mllib\_ImageConvolveMxN, 409  
MxN convolution, with kernel analysis for  
taking advantage of special cases —  
  mllib\_ImageConvolveMxN\_Fp, 411  
MxN convolution on a color indexed image —  
  mllib\_ImageConvMxNIndex, 407  
MxN gradient filter —  
  mllib\_ImageGradientMxN, 485  
MxN gradient filter —  
  mllib\_ImageGradientMxN\_Fp, 487  
MxN median filter —  
  mllib\_ImageMedianFilterMxN, 573  
MxN median filter —  
  mllib\_ImageMedianFilterMxN\_Fp, 575  
MxN median filter, unsigned —  
  mllib\_ImageMedianFilterMxN\_US, 577  
MxN rank filter —  
  mllib\_ImageRankFilterMxN, 650  
MxN rank filter —  
  mllib\_ImageRankFilterMxN\_Fp, 652  
MxN rank filter, unsigned —  
  mllib\_ImageRankFilterMxN\_US, 654  
  
**N**  
Not — mllib\_ImageNot, 607

Not — `mlib_ImageNot_Inp`, 610  
 NotAnd — `mlib_ImageNotAnd`, 608  
 NotAnd, in place —  
     `mlib_ImageNotAnd_Inp`, 609  
 NotAnd with a constant —  
     `mlib_ImageConstNotAnd`, 355  
 NotAnd with a constant, in place —  
     `mlib_ImageConstNotAnd_Inp`, 356  
 NotOr — `mlib_ImageNotOr`, 611  
 NotOr, in place — `mlib_ImageNotOr_Inp`, 612  
 NotOr with a constant —  
     `mlib_ImageConstNotOr`, 357  
 NotOr with a constant, in place —  
     `mlib_ImageConstNotOr_Inp`, 358  
 NotXor — `mlib_ImageNotXor`, 613  
 NotXor, in place —  
     `mlib_ImageNotXor_Inp`, 614  
 NotXor with a constant —  
     `mlib_ImageConstNotXor`, 359  
 NotXor with a constant, in place —  
     `mlib_ImageConstNotXor_Inp`, 360

## O

Or — `mlib_ImageOr`, 615  
 Or, in place — `mlib_ImageOr_Inp`, 616  
 Or with a constant — `mlib_ImageConstOr`, 361  
 Or with a constant, in place —  
     `mlib_ImageConstOr_Inp`, 362  
 OrNot — `mlib_ImageOrNot`, 619  
 OrNot, in place — `mlib_ImageOrNot1_Inp`, 617  
 OrNot, in place — `mlib_ImageOrNot2_Inp`, 618  
 OrNot with a constant —  
     `mlib_ImageConstOrNot`, 363  
 OrNot with a constant, in place —  
     `mlib_ImageConstOrNot_Inp`, 364

## P

parallel Infinite Impulse Response (IIR) filtering  
     — `mlib_SignalIIRFree_P4_F32S_F32S`, 1029  
 parallel Infinite Impulse Response (IIR) filtering  
     — `mlib_SignalIIRFree_P4_F32_F32`, 1029  
 parallel Infinite Impulse Response (IIR) filtering  
     — `mlib_SignalIIRFree_P4_S16S_S16S`, 1030

parallel Infinite Impulse Response (IIR) filtering  
     — `mlib_SignalIIRFree_P4_S16_S16`, 1030  
 parallel Infinite Impulse Response (IIR) filtering  
     — `mlib_SignalIIRInit_P4_F32S_F32S`, 1033  
 parallel Infinite Impulse Response (IIR) filtering  
     — `mlib_SignalIIRInit_P4_F32_F32`, 1033  
 parallel Infinite Impulse Response (IIR) filtering  
     — `mlib_SignalIIRInit_P4_S16S_S16S`, 1034  
 parallel Infinite Impulse Response (IIR) filtering  
     — `mlib_SignalIIRInit_P4_S16_S16`, 1034  
 parallel Infinite Impulse Response (IIR) filtering  
     — `mlib_SignalIIR_P4_F32S_F32S`, 1035  
 parallel Infinite Impulse Response (IIR) filtering  
     — `mlib_SignalIIR_P4_F32_F32`, 1035  
 parallel Infinite Impulse Response (IIR) filtering  
     — `mlib_SignalIIR_P4_S16S_S16S_Sat`, 1035  
 parallel Infinite Impulse Response (IIR) filtering  
     — `mlib_SignalIIR_P4_S16_S16_Sat`, 1035  
 perform linear predictive coding with  
     autocorrelation method —  
         `mlib_SignalLPCAutoCorrel_S16`, 1078  
 perform linear predictive coding with  
     autocorrelation method —  
         `mlib_SignalLPCAutoCorrel_S16_AdP`, 1078  
 perform linear predictive coding with  
     covariance method —  
         `mlib_SignalLPC_Covariance_S16`, 1084  
 perform linear predictive coding with  
     covariance method —  
         `mlib_SignalLPC_Covariance_S16_AdP`, 1084  
 perform cepstral analysis —  
     `mlib_SignalCepstral_F32`, 859  
 perform cepstral analysis —  
     `mlib_SignalCepstral_S16`, 863  
 perform cepstral analysis —  
     `mlib_SignalCepstral_S16_AdP`, 865  
 perform cepstral analysis in mel frequency scale  
     — `mlib_SignalMelCepstral_F32`, 1099  
 perform cepstral analysis in mel frequency scale  
     — `mlib_SignalMelCepstral_S16`, 1104  
 perform cepstral analysis in mel frequency scale  
     — `mlib_SignalMelCepstral_S16_AdP`, 1106  
 perform dynamic time warping for K-best paths  
     on scalar data —  
         `mlib_SignalDTW_K_Scalar_F32`, 879  
 perform dynamic time warping for K-best paths  
     on scalar data —  
         `mlib_SignalDTW_K_Scalar_S16`, 890

- perform dynamic time warping for K-best paths on vector data —
  - mllib\_SignalDTWKVector\_F32, 894
- perform dynamic time warping for K-best paths on vector data —
  - mllib\_SignalDTWKVector\_S16, 907
- perform dynamic time warping on scalar data —
  - mllib\_SignalDTWScalarPath\_F32, 918
- perform dynamic time warping on scalar data —
  - mllib\_SignalDTWScalarPath\_S16, 922
- perform dynamic time warping on scalar data —
  - mllib\_SignalDTWScalar\_F32, 911
- perform dynamic time warping on scalar data —
  - mllib\_SignalDTWScalar\_S16, 926
- perform dynamic time warping on vector data —
  - mllib\_SignalDTWVectorPath\_F32, 938
- perform dynamic time warping on vector data —
  - mllib\_SignalDTWVectorPath\_S16, 942
- perform dynamic time warping on vector data —
  - mllib\_SignalDTWVector\_F32, 930
- perform dynamic time warping on vector data —
  - mllib\_SignalDTWVector\_S16, 946
- perform linear predictive coding with autocorrelation method —
  - mllib\_SignalLPCAutoCorrel\_F32, 1066
- perform linear predictive coding with covariance method —
  - mllib\_SignalLPCCovariance\_F32, 1080
- perform open-loop pitch analysis —
  - mllib\_SignalLPCPitchAnalyze\_F32, 1091
- perform open-loop pitch analysis —
  - mllib\_SignalLPCPitchAnalyze\_S16, 1093
- perform perceptual weighting on input signal —
  - mllib\_SignalLPCPerceptWeight\_F32, 1086
- perform perceptual weighting on input signal —
  - mllib\_SignalLPCPerceptWeight\_S16, 1089
- polynomial-based image warp —
  - mllib\_ImagePolynomialWarp, 620
- polynomial-based image warp —
  - mllib\_ImagePolynomialWarp\_Fp, 623
- polynomial-based image warp with table-driven interpolation —
  - mllib\_ImagePolynomialWarpTable, 626
- polynomial-based image warp with table-driven interpolation —
  - mllib\_ImagePolynomialWarpTable\_Fp, 629

## Q

- quantization of forward Discrete Cosine Transform (DCT) coefficients —
  - mllib\_VideoQuantizeInit\_S16, 1517
- quantization of forward Discrete Cosine Transform (DCT) coefficients —
  - mllib\_VideoQuantize\_S16, 1518

## R

- reallocate a block of bytes —
  - mllib\_realloc, 848
- rectangular windowing multiplication —
  - mllib\_SignalMulRectangular\_F32, 1132
- rectangular windowing multiplication —
  - mllib\_SignalMulRectangular\_F32S, 1132
- rectangular windowing multiplication —
  - mllib\_SignalMulRectangular\_F32S\_F32S, 1133
- rectangular windowing multiplication —
  - mllib\_SignalMulRectangular\_F32\_F32, 1133
- rectangular windowing multiplication —
  - mllib\_SignalMulRectangular\_S16, 1134
- rectangular windowing multiplication —
  - mllib\_SignalMulRectangular\_S16S, 1134
- rectangular windowing multiplication —
  - mllib\_SignalMulRectangular\_S16S\_S16S, 1135
- rectangular windowing multiplication —
  - mllib\_SignalMulRectangular\_S16\_S16, 1135
- release the internal data structure for image dithering —
  - mllib\_ImageColorDitherFree, 293
- releases the internal data structure for true color to indexed color conversion —
  - mllib\_ImageColorTrue2IndexFree, 324
- replace a color in an image —
  - mllib\_ImageReplaceColor, 659
- replace a color in an image —
  - mllib\_ImageReplaceColor\_Fp, 660
- replace a color in an image, in place —
  - mllib\_ImageReplaceColor\_Fp\_Inp, 661
- replace a color in an image, in place —
  - mllib\_ImageReplaceColor\_Inp, 662
- resampling with filtering —
  - mllib\_SignalReSampleFIRFree\_S16S\_S16S, 1169
- resampling with filtering —
  - mllib\_SignalReSampleFIRFree\_S16\_S16, 1169
- resampling with filtering —
  - mllib\_SignalReSampleFIR\_S16S\_S16S\_Sat, 1172

resampling with filtering —  
     mllib\_SignalReSampleFIR\_S16\_S16\_Sat, 1172  
 resampling with filtering —  
     mllib\_SignalReSampleFIRFree\_F32S\_F32S, 1168  
 resampling with filtering —  
     mllib\_SignalReSampleFIRFree\_F32\_F32, 1167  
 resampling with filtering —  
     mllib\_SignalReSampleFIR\_F32S\_F32S, 1166  
 resampling with filtering —  
     mllib\_SignalReSampleFIR\_F32\_F32, 1165  
 return K-best path on scalar data —  
     mllib\_SignalDTWKScalarPath\_F32, 886  
 return K-best path on scalar data —  
     mllib\_SignalDTWKScalarPath\_S16, 886  
 return K-best path on vector data —  
     mllib\_SignalDTWKVectorPath\_F32, 903  
 return K-best path on vector data —  
     mllib\_SignalDTWKVectorPath\_S16, 903  
 return the energy of the input signal —  
     mllib\_SignalLPCAutoCorrelGetEnergy\_S16  
     \_Adp, 1071  
 return the energy of the input signal —  
     mllib\_SignalLPCAutoCorrelGetEnergy\_S16, 1071  
 return the partial correlation (PARCOR)  
     coefficients —  
     mllib\_SignalLPCAutoCorrelGetPARCOR\_S16, 1075  
 return the partial correlation (PARCOR)  
     coefficients —  
     mllib\_SignalLPCAutoCorrelGetPARCOR\_S16  
     \_Adp, 1075  
 return a version string — mllib\_version, 1288  
 return the energy of the input signal —  
     mllib\_SignalLPCAutoCorrelGetEnergy\_F32, 1069  
 return the partial correlation (PARCOR)  
     coefficients —  
     mllib\_SignalLPCAutoCorrelGetPARCOR\_F32, 1073  
 reverse byte order of vector —  
     mllib\_VectorReverseByteOrder\_D64\_D64, 1265  
 reverse byte order of vector —  
     mllib\_VectorReverseByteOrder\_F32\_F32, 1265  
 reverse byte order of vector —  
     mllib\_VectorReverseByteOrder\_S16\_S16, 1265  
 reverse byte order of vector —  
     mllib\_VectorReverseByteOrder\_S32\_S32, 1265  
 reverse byte order of vector —  
     mllib\_VectorReverseByteOrder\_S64\_S64, 1265  
 reverse byte order of vector —  
     mllib\_VectorReverseByteOrder\_U16\_U16, 1265  
 reverse byte order of vector —  
     mllib\_VectorReverseByteOrder\_U32\_U32, 1265  
 reverse byte order of vector —  
     mllib\_VectorReverseByteOrder\_U64\_U64, 1265  
 reverse byte order of vector, in place —  
     mllib\_VectorReverseByteOrder\_D64, 1263  
 reverse byte order of vector, in place —  
     mllib\_VectorReverseByteOrder\_F32, 1263  
 reverse byte order of vector, in place —  
     mllib\_VectorReverseByteOrder\_S16, 1263  
 reverse byte order of vector, in place —  
     mllib\_VectorReverseByteOrder\_S32, 1263  
 reverse byte order of vector, in place —  
     mllib\_VectorReverseByteOrder\_S64, 1263  
 reverse byte order of vector, in place —  
     mllib\_VectorReverseByteOrder\_U16, 1263  
 reverse byte order of vector, in place —  
     mllib\_VectorReverseByteOrder\_U32, 1263  
 reverse byte order of vector, in place —  
     mllib\_VectorReverseByteOrder\_U64, 1263  
 reverse byte order of vector —  
     mllib\_VectorReverseByteOrder, 1261  
 reverse byte order of vector, in place —  
     mllib\_VectorReverseByteOrder\_Inp, 1262  
 reversible color space conversion for wavelet  
     transformation —  
     mllib\_VideoReversibleColorRGB2YUV\_  
     S16\_S16, 1519  
 reversible color space conversion for wavelet  
     transformation —  
     mllib\_VideoReversibleColorRGB2YUV\_  
     S16\_U8, 1519  
 reversible color space conversion for wavelet  
     transformation —  
     mllib\_VideoReversibleColorRGB2YUV\_  
     S32\_S16, 1519  
 reversible color space conversion for wavelet  
     transformation —  
     mllib\_VideoReversibleColorRGB2YUV\_  
     U8\_U8, 1519  
 reversible color space conversion for wavelet  
     transformation —  
     mllib\_VideoReversibleColorYUV2RGB\_  
     S16\_S16, 1521  
 reversible color space conversion for wavelet  
     transformation —  
     mllib\_VideoReversibleColorYUV2RGB\_  
     S16\_S32, 1521

- reversible color space conversion for wavelet transformation —
  - mllib\_VideoReversibleColorYUV2RGB\_U8\_S16, 1521
- reversible color space conversion for wavelet transformation —
  - mllib\_VideoReversibleColorYUV2RGB\_U8\_U8, 1521
- RGB to HSL color conversion —
  - mllib\_ImageColorRGB2HSL, 311
- RGB to HSL color conversion —
  - mllib\_ImageColorRGB2HSL\_Fp, 313
- RGB to HSV color conversion —
  - mllib\_ImageColorRGB2HSV, 314
- RGB to HSV color conversion —
  - mllib\_ImageColorRGB2HSV\_Fp, 316
- RGB to JFIF YCbCr color conversion —
  - mllib\_VideoColorRGB2JFIFYCC420, 1327
- RGB to JFIF YCbCr color conversion —
  - mllib\_VideoColorRGB2JFIFYCC422, 1328
- RGB to JFIF YCbCr color conversion —
  - mllib\_VideoColorRGB2JFIFYCC444, 1329
- RGB to JFIF YCbCr color conversion —
  - mllib\_VideoColorRGB2JFIFYCC444\_S16, 1330
- RGB to monochrome conversion —
  - mllib\_ImageColorRGB2CIEMono, 309
- RGB to monochrome conversion —
  - mllib\_ImageColorRGB2CIEMono\_Fp, 310
- RGB to monochrome conversion —
  - mllib\_ImageColorRGB2Mono, 317
- RGB to monochrome conversion —
  - mllib\_ImageColorRGB2Mono\_Fp, 318
- RGB to XYZ color conversion —
  - mllib\_ImageColorRGB2XYZ, 319
- RGB to XYZ color conversion —
  - mllib\_ImageColorRGB2XYZ\_Fp, 320
- RGB to YCC color conversion —
  - mllib\_ImageColorRGB2YCC, 321
- RGB to YCC color conversion —
  - mllib\_ImageColorRGB2YCC\_Fp, 322
- rotate an image by 180 degrees —
  - mllib\_ImageRotate180, 663
- rotate an image by 180 degrees —
  - mllib\_ImageRotate180\_Fp, 664
- rotate an image by 270 degrees —
  - mllib\_ImageRotate270, 665
- rotate an image by 270 degrees —
  - mllib\_ImageRotate270\_Fp, 666

- rotate an image by 90 degrees —
  - mllib\_ImageRotate90, 669
- rotate an image by 90 degrees —
  - mllib\_ImageRotate90\_Fp, 670
- rotate color-indexed image —
  - mllib\_ImageRotateIndex, 673
- rotate image — mllib\_ImageRotate, 667
- rotate image — mllib\_ImageRotate\_Fp, 671

## S

- second moment — mllib\_ImageMoment2, 597
- second moment —
  - mllib\_ImageMoment2\_Fp, 598
- separable 3x3 convolution —
  - mllib\_ImageSConv3x3, 688
- separable 3x3 convolution —
  - mllib\_ImageSConv3x3\_Fp, 690
- separable 5x5 convolution —
  - mllib\_ImageSConv5x5, 692
- separable 5x5 convolution —
  - mllib\_ImageSConv5x5\_Fp, 694
- separable 7x7 convolution —
  - mllib\_ImageSConv7x7, 696
- separable 7x7 convolution —
  - mllib\_ImageSConv7x7\_Fp, 698
- set vector to specified value —
  - mllib\_VectorSet\_S16, 1272
- set vector to specified value —
  - mllib\_VectorSet\_S16C, 1272
- set vector to specified value —
  - mllib\_VectorSet\_S32, 1272
- set vector to specified value —
  - mllib\_VectorSet\_S32C, 1272
- set vector to specified value —
  - mllib\_VectorSet\_S8, 1272
- set vector to specified value —
  - mllib\_VectorSet\_S8C, 1272
- set vector to specified value —
  - mllib\_VectorSet\_U8, 1272
- set vector to specified value —
  - mllib\_VectorSet\_U8C, 1272
- set a block of bytes — mllib\_memset, 847
- set format — mllib\_ImageSetFormat, 701
- set paddings — mllib\_ImageSetPaddings, 702
- sets edges of an image to a specific color —
  - mllib\_ImageClearEdge, 286

sets edges of an image to a specific color —  
 mlib\_ImageClearEdge\_Fp, 287

signal auto-correlation —  
 mlib\_SignalAutoCorrel\_F32, 857

signal auto-correlation —  
 mlib\_SignalAutoCorrel\_F32S, 857

signal auto-correlation —  
 mlib\_SignalAutoCorrel\_S16, 857

signal auto-correlation —  
 mlib\_SignalAutoCorrel\_S16S, 857

signal convolution —  
 mlib\_SignalConv\_F32S\_F32S, 873

signal convolution —  
 mlib\_SignalConv\_F32\_F32, 873

signal convolution —  
 mlib\_SignalConv\_S16S\_S16S\_Sat, 873

signal convolution —  
 mlib\_SignalConv\_S16\_S16\_Sat, 873

signal cross correlation —  
 mlib\_SignalCrossCorrel\_F32, 875

signal cross correlation —  
 mlib\_SignalCrossCorrel\_F32S, 875

signal cross correlation —  
 mlib\_SignalCrossCorrel\_S16, 875

signal cross correlation —  
 mlib\_SignalCrossCorrel\_S16S, 875

signal downsampling —  
 mlib\_SignalDownSample\_F32S\_F32S, 877

signal downsampling —  
 mlib\_SignalDownSample\_F32\_F32, 877

signal downsampling —  
 mlib\_SignalDownSample\_S16S\_S16S, 877

signal downsampling —  
 mlib\_SignalDownSample\_S16\_S16, 877

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_1, 954

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_1\_D64, 954

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_1\_D64C, 954

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_1\_D64C\_D64, 954

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_1\_D64C\_D64C, 954

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_1\_D64\_D64, 954

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_1\_F32, 954

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_1\_F32C, 954

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_1\_F32C\_F32, 954

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_1\_F32C\_F32C, 954

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_1\_F32\_F32, 954

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_1\_S16C\_Mod, 954

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_1\_S16C\_S16C\_Mod, 954

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_1\_S16C\_S16\_Mod, 954

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_1\_S16\_Mod, 954

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_1\_S16\_S16\_Mod, 954

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_2, 957

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_2\_D64, 957

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_2\_D64C, 957

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_2\_D64C\_D64, 957

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_2\_D64C\_D64C, 957

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_2\_D64\_D64, 957

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_2\_F32, 957

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_2\_F32C, 957

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_2\_F32C\_F32, 957

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_2\_F32C\_F32C, 957

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_2\_F32\_F32, 957

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_2\_S16, 957

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_2\_S16C, 957

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_2\_S16C\_S16, 957

signal Fast Fourier Transform (FFT) —  
 mlib\_SignalFFT\_2\_S16C\_S16C, 957

signal Fast Fourier Transform (FFT) —  
     mllib\_SignalFFT\_2\_S16\_S16, 957  
 signal Fast Fourier Transform (FFT) —  
     mllib\_SignalFFT\_3, 960  
 signal Fast Fourier Transform (FFT) —  
     mllib\_SignalFFT\_3\_D64, 960  
 signal Fast Fourier Transform (FFT) —  
     mllib\_SignalFFT\_3\_D64C, 960  
 signal Fast Fourier Transform (FFT) —  
     mllib\_SignalFFT\_3\_D64C\_D64, 960  
 signal Fast Fourier Transform (FFT) —  
     mllib\_SignalFFT\_3\_D64C\_D64C, 960  
 signal Fast Fourier Transform (FFT) —  
     mllib\_SignalFFT\_3\_D64\_D64, 960  
 signal Fast Fourier Transform (FFT) —  
     mllib\_SignalFFT\_3\_F32, 960  
 signal Fast Fourier Transform (FFT) —  
     mllib\_SignalFFT\_3\_F32C, 960  
 signal Fast Fourier Transform (FFT) —  
     mllib\_SignalFFT\_3\_F32C\_F32, 960  
 signal Fast Fourier Transform (FFT) —  
     mllib\_SignalFFT\_3\_F32C\_F32C, 960  
 signal Fast Fourier Transform (FFT) —  
     mllib\_SignalFFT\_3\_F32\_F32, 960  
 signal Fast Fourier Transform (FFT) —  
     mllib\_SignalFFT\_3\_S16C\_Mod, 960  
 signal Fast Fourier Transform (FFT) —  
     mllib\_SignalFFT\_3\_S16C\_S16C\_Mod, 960  
 signal Fast Fourier Transform (FFT) —  
     mllib\_SignalFFT\_3\_S16C\_S16\_Mod, 960  
 signal Fast Fourier Transform (FFT) —  
     mllib\_SignalFFT\_3\_S16\_Mod, 960  
 signal Fast Fourier Transform (FFT) —  
     mllib\_SignalFFT\_3\_S16\_S16\_Mod, 960  
 signal Fast Fourier Transform (FFT) —  
     mllib\_SignalFFT\_4, 963  
 signal Fast Fourier Transform (FFT) —  
     mllib\_SignalFFT\_4\_S16, 963  
 signal Fast Fourier Transform (FFT) —  
     mllib\_SignalFFT\_4\_S16C, 963  
 signal Fast Fourier Transform (FFT) —  
     mllib\_SignalFFT\_4\_S16C\_S16, 963  
 signal Fast Fourier Transform (FFT) —  
     mllib\_SignalFFT\_4\_S16C\_S16C, 963  
 signal Fast Fourier Transform (FFT) —  
     mllib\_SignalFFT\_4\_S16\_S16, 963  
 signal Fast Fourier Transform with windowing  
 (FFTW) — mllib\_SignalFFTW\_1, 965  
 signal Fast Fourier Transform with windowing  
 (FFTW) — mllib\_SignalFFTW\_1\_F32, 965  
 signal Fast Fourier Transform with windowing  
 (FFTW) — mllib\_SignalFFTW\_1\_F32C, 965  
 signal Fast Fourier Transform with windowing  
 (FFTW) —  
     mllib\_SignalFFTW\_1\_F32C\_F32, 965  
 signal Fast Fourier Transform with windowing  
 (FFTW) —  
     mllib\_SignalFFTW\_1\_F32C\_F32C, 965  
 signal Fast Fourier Transform with windowing  
 (FFTW) —  
     mllib\_SignalFFTW\_1\_F32\_F32, 965  
 signal Fast Fourier Transform with windowing  
 (FFTW) —  
     mllib\_SignalFFTW\_1\_S16C\_Mod, 965  
 signal Fast Fourier Transform with windowing  
 (FFTW) —  
     mllib\_SignalFFTW\_1\_S16C\_S16C\_Mod, 965  
 signal Fast Fourier Transform with windowing  
 (FFTW) —  
     mllib\_SignalFFTW\_1\_S16C\_S16\_Mod, 965  
 signal Fast Fourier Transform with windowing  
 (FFTW) —  
     mllib\_SignalFFTW\_1\_S16\_Mod, 965  
 signal Fast Fourier Transform with windowing  
 (FFTW) —  
     mllib\_SignalFFTW\_1\_S16\_S16\_Mod, 965  
 signal Fast Fourier Transform with windowing  
 (FFTW) — mllib\_SignalFFTW\_2, 968  
 signal Fast Fourier Transform with windowing  
 (FFTW) — mllib\_SignalFFTW\_2\_F32, 968  
 signal Fast Fourier Transform with windowing  
 (FFTW) — mllib\_SignalFFTW\_2\_F32C, 968  
 signal Fast Fourier Transform with windowing  
 (FFTW) —  
     mllib\_SignalFFTW\_2\_F32C\_F32, 968  
 signal Fast Fourier Transform with windowing  
 (FFTW) —  
     mllib\_SignalFFTW\_2\_F32C\_F32C, 968  
 signal Fast Fourier Transform with windowing  
 (FFTW) —  
     mllib\_SignalFFTW\_2\_F32\_F32, 968  
 signal Fast Fourier Transform with windowing  
 (FFTW) — mllib\_SignalFFTW\_2\_S16, 968  
 signal Fast Fourier Transform with windowing  
 (FFTW) — mllib\_SignalFFTW\_2\_S16C, 968



signal Fast Fourier Transform with windowing (FFTW) —  
     mllib\_SignalFFTW\_2\_S16C\_S16, 968  
 signal Fast Fourier Transform with windowing (FFTW) —  
     mllib\_SignalFFTW\_2\_S16C\_S16C, 968  
 signal Fast Fourier Transform with windowing (FFTW) —  
     mllib\_SignalFFTW\_2\_S16\_S16, 968  
 signal Fast Fourier Transform with windowing (FFTW) — mllib\_SignalFFTW\_3, 971  
 signal Fast Fourier Transform with windowing (FFTW) — mllib\_SignalFFTW\_3\_F32, 971  
 signal Fast Fourier Transform with windowing (FFTW) — mllib\_SignalFFTW\_3\_F32C, 971  
 signal Fast Fourier Transform with windowing (FFTW) —  
     mllib\_SignalFFTW\_3\_F32C\_F32, 971  
 signal Fast Fourier Transform with windowing (FFTW) —  
     mllib\_SignalFFTW\_3\_F32C\_F32C, 971  
 signal Fast Fourier Transform with windowing (FFTW) —  
     mllib\_SignalFFTW\_3\_F32\_F32, 971  
 signal Fast Fourier Transform with windowing (FFTW) —  
     mllib\_SignalFFTW\_3\_S16C\_Mod, 971  
 signal Fast Fourier Transform with windowing (FFTW) —  
     mllib\_SignalFFTW\_3\_S16C\_S16C\_Mod, 971  
 signal Fast Fourier Transform with windowing (FFTW) —  
     mllib\_SignalFFTW\_3\_S16C\_S16\_Mod, 971  
 signal Fast Fourier Transform with windowing (FFTW) —  
     mllib\_SignalFFTW\_3\_S16\_Mod, 971  
 signal Fast Fourier Transform with windowing (FFTW) —  
     mllib\_SignalFFTW\_3\_S16\_S16\_Mod, 971  
 signal Fast Fourier Transform with windowing (FFTW) — mllib\_SignalFFTW\_4, 974  
 signal Fast Fourier Transform with windowing (FFTW) — mllib\_SignalFFTW\_4\_S16, 974  
 signal Fast Fourier Transform with windowing (FFTW) — mllib\_SignalFFTW\_4\_S16C, 974  
 signal Fast Fourier Transform with windowing (FFTW) —  
     mllib\_SignalFFTW\_4\_S16C\_S16, 974  
 signal Fast Fourier Transform with windowing (FFTW) —  
     mllib\_SignalFFTW\_4\_S16C\_S16C, 974  
 signal Fast Fourier Transform with windowing (FFTW) —  
     mllib\_SignalFFTW\_4\_S16\_S16, 974  
 signal hard limiting — mllib\_SignalLimit, 1042  
 signal hard limiting —  
     mllib\_SignalLimit\_F32, 1042  
 signal hard limiting —  
     mllib\_SignalLimit\_F32S, 1042  
 signal hard limiting —  
     mllib\_SignalLimit\_F32S\_F32S, 1042  
 signal hard limiting —  
     mllib\_SignalLimit\_F32\_F32, 1042  
 signal hard limiting —  
     mllib\_SignalLimit\_S16, 1042  
 signal hard limiting —  
     mllib\_SignalLimit\_S16S, 1042  
 signal hard limiting —  
     mllib\_SignalLimit\_S16S\_S16S, 1042  
 signal hard limiting —  
     mllib\_SignalLimit\_S16\_S16, 1042  
 signal Inverse Fast Fourier Transform (IFFT) —  
     mllib\_SignalIFFT\_1, 1002  
 signal Inverse Fast Fourier Transform (IFFT) —  
     mllib\_SignalIFFT\_1\_D64, 1002  
 signal Inverse Fast Fourier Transform (IFFT) —  
     mllib\_SignalIFFT\_1\_D64C, 1002  
 signal Inverse Fast Fourier Transform (IFFT) —  
     mllib\_SignalIFFT\_1\_D64C\_D64C, 1002  
 signal Inverse Fast Fourier Transform (IFFT) —  
     mllib\_SignalIFFT\_1\_D64\_D64, 1002  
 signal Inverse Fast Fourier Transform (IFFT) —  
     mllib\_SignalIFFT\_1\_D64\_D64C, 1002  
 signal Inverse Fast Fourier Transform (IFFT) —  
     mllib\_SignalIFFT\_1\_F32, 1002  
 signal Inverse Fast Fourier Transform (IFFT) —  
     mllib\_SignalIFFT\_1\_F32C, 1002  
 signal Inverse Fast Fourier Transform (IFFT) —  
     mllib\_SignalIFFT\_1\_F32C\_F32C, 1002  
 signal Inverse Fast Fourier Transform (IFFT) —  
     mllib\_SignalIFFT\_1\_F32\_F32, 1002  
 signal Inverse Fast Fourier Transform (IFFT) —  
     mllib\_SignalIFFT\_1\_F32\_F32C, 1002  
 signal Inverse Fast Fourier Transform (IFFT) —  
     mllib\_SignalIFFT\_1\_S16, 1002





signal Inverse Fast Fourier Transform with windowing (IFFTW) —  
     mllib\_SignalIFFTW\_4\_S16C\_S16C, 1022  
 signal Inverse Fast Fourier Transform with windowing (IFFTW) —  
     mllib\_SignalIFFTW\_4\_S16\_S16, 1022  
 signal Inverse Fast Fourier Transform with windowing (IFFTW) —  
     mllib\_SignalIFFTW\_4\_S16\_S16C, 1022  
 signal pre-emphasizing —  
     mllib\_SignalEmphasize\_F32S\_F32S, 952  
 signal pre-emphasizing —  
     mllib\_SignalEmphasize\_F32\_F32, 952  
 signal pre-emphasizing —  
     mllib\_SignalEmphasize\_S16S\_S16S\_Sat, 952  
 signal pre-emphasizing —  
     mllib\_SignalEmphasize\_S16\_S16\_Sat, 952  
 signal upsampling —  
     mllib\_SignalUpSample\_F32S\_F32S, 1192  
 signal upsampling —  
     mllib\_SignalUpSample\_F32\_F32, 1192  
 signal upsampling —  
     mllib\_SignalUpSample\_S16S\_S16S, 1192  
 signal upsampling —  
     mllib\_SignalUpSample\_S16\_S16, 1192  
 sine wave generation —  
     mllib\_SignalSineWaveFree\_F32, 1174  
 sine wave generation —  
     mllib\_SignalSineWaveFree\_S16, 1175  
 sine wave generation —  
     mllib\_SignalSineWaveInit\_F32, 1176  
 sine wave generation —  
     mllib\_SignalSineWaveInit\_S16, 1177  
 sine wave generation —  
     mllib\_SignalSineWave\_F32, 1173  
 sine wave generation —  
     mllib\_SignalSineWave\_S16, 1178  
 Sobel filter — mllib\_ImageSobel, 704  
 Sobel filter — mllib\_ImageSobel\_Fp, 704  
 split — mllib\_SignalSplit\_F32\_F32S, 1179  
 split — mllib\_SignalSplit\_S16\_S16S, 1180  
 square — mllib\_ImageSqr\_Fp, 706  
 square, in place — mllib\_ImageSqr\_Fp\_Inp, 707  
 square with shifting —  
     mllib\_ImageSqrShift, 708  
 square with shifting, in place —  
     mllib\_ImageSqrShift\_Inp, 709  
 subimage creation —  
     mllib\_ImageCreateSubimage, 420  
 subsamples an image with a box filter —  
     mllib\_ImageSubsampleAverage, 718  
 subsamples an image with a box filter —  
     mllib\_ImageSubsampleAverage\_Fp, 718  
 subsamples a binary image and converts it to a grayscale image —  
     mllib\_ImageSubsampleBinaryToGray, 720  
 subtraction — mllib\_ImageSub, 716  
 subtraction — mllib\_ImageSub\_Fp, 717  
 subtraction, in place —  
     mllib\_ImageSub1\_Fp\_Inp, 712  
 subtraction, in place —  
     mllib\_ImageSub1\_Inp, 713  
 subtraction, in place —  
     mllib\_ImageSub2\_Fp\_Inp, 714  
 subtraction, in place —  
     mllib\_ImageSub2\_Inp, 715  
 Subtraction with a constant —  
     mllib\_ImageConstSub, 365  
 Subtraction with a constant —  
     mllib\_ImageConstSub\_Fp, 366  
 subtraction with a constant —  
     mllib\_ImageConstSub\_Fp\_Inp, 367  
 Subtraction with a constant —  
     mllib\_ImageConstSub\_Inp, 368  
 sum of the absolute values of a vector —  
     mllib\_VectorSumAbs\_S16\_Sat, 1286  
 sum of the absolute values of a vector —  
     mllib\_VectorSumAbs\_S32\_Sat, 1286  
 sum of the absolute values of a vector —  
     mllib\_VectorSumAbs\_S8\_Sat, 1286  
 sum of the absolute values of a vector —  
     mllib\_VectorSumAbs\_U8\_Sat, 1286  
 sum of the absolute values of the differences of two vectors —  
     mllib\_VectorSumAbsDiff\_S16\_Sat, 1285  
 sum of the absolute values of the differences of two vectors —  
     mllib\_VectorSumAbsDiff\_S32\_Sat, 1285  
 sum of the absolute values of the differences of two vectors —  
     mllib\_VectorSumAbsDiff\_S8\_Sat, 1285  
 sum of the absolute values of the differences of two vectors —  
     mllib\_VectorSumAbsDiff\_U8\_Sat, 1285

## T

- table lookup — `mlib_ImageLookUp2`, 528
- table lookup — `mlib_ImageLookUp`, 530
- table lookup, in place —
  - `mlib_ImageLookUp_Inp`, 532
- table lookup with mask —
  - `mlib_ImageLookUpMask`, 533
- test flags — `mlib_ImageTestFlags`, 722
- true color to indexed color conversion using error diffusion —
  - `mlib_ImageColorErrorDiffusion3x3`, 297
- true color to indexed color conversion using ordered dithering —
  - `mlib_ImageColorOrderedDither8x8`, 306
- true-color to indexed-color or grayscale to black-white conversion, using error diffusion — `mlib_ImageColorErrorDiffusionMxN`, 298
- true-color to indexed-color or grayscale to black-white conversion, using ordered dithering —
  - `mlib_ImageColorOrderedDitherMxN`, 307
- true color to indexed color using nearest matched LUT entries —
  - `mlib_ImageColorTrue2Index`, 323
- two image maximum — `mlib_ImageMax`, 535
- two image maximum —
  - `mlib_ImageMax_Fp`, 548
- two image maximum —
  - `mlib_ImageMax_Fp_Inp`, 549
- two image maximum —
  - `mlib_ImageMax_Inp`, 552
- two-image minimum — `mlib_ImageMin`, 579
- two-image minimum —
  - `mlib_ImageMin_Fp`, 592
- two-image minimum —
  - `mlib_ImageMin_Fp_Inp`, 593
- two-image minimum —
  - `mlib_ImageMin_Inp`, 596

## U

- Unit matrix generation —
  - `mlib_MatrixUnit_S16`, 843
- Unit matrix generation —
  - `mlib_MatrixUnit_S16C`, 843
- Unit matrix generation —
  - `mlib_MatrixUnit_S32`, 843

- Unit matrix generation —
  - `mlib_MatrixUnit_S32C`, 843
- Unit matrix generation —
  - `mlib_MatrixUnit_S8`, 843
- Unit matrix generation —
  - `mlib_MatrixUnit_S8C`, 843
- Unit matrix generation —
  - `mlib_MatrixUnit_U8`, 843
- Unit matrix generation —
  - `mlib_MatrixUnit_U8C`, 843
- up sampling rate conversion in JFIF —
  - `mlib_VideoUpSample420`, 1528
- up sampling rate conversion in JFIF —
  - `mlib_VideoUpSample420_Nearest`, 1529
- up sampling rate conversion in JFIF —
  - `mlib_VideoUpSample420_Nearest_S16`, 1530
- up sampling rate conversion in JFIF —
  - `mlib_VideoUpSample420_S16`, 1531
- up sampling rate conversion in JFIF —
  - `mlib_VideoUpSample422`, 1532
- up sampling rate conversion in JFIF —
  - `mlib_VideoUpSample422_Nearest`, 1533
- up sampling rate conversion in JFIF —
  - `mlib_VideoUpSample422_Nearest_S16`, 1534
- up sampling rate conversion in JFIF —
  - `mlib_VideoUpSample422_S16`, 1535
- upsampling with filtering —
  - `mlib_SignalUpSampleFIRFree_S16S_S16S`, 1187
- upsampling with filtering —
  - `mlib_SignalUpSampleFIRFree_S16_S16`, 1187
- upsampling with filtering —
  - `mlib_SignalUpSampleFIRInit_S16S_S16S`, 1190
- upsampling with filtering —
  - `mlib_SignalUpSampleFIRInit_S16_S16`, 1190
- upsampling with filtering —
  - `mlib_SignalUpSampleFIR_S16S_S16S_Sat`, 1191
- upsampling with filtering —
  - `mlib_SignalUpSampleFIR_S16_S16_Sat`, 1191
- upsampling with filtering —
  - `mlib_SignalUpSampleFIRFree_F32S_F32S`, 1186
- upsampling with filtering —
  - `mlib_SignalUpSampleFIRFree_F32_F32`, 1185
- upsampling with filtering —
  - `mlib_SignalUpSampleFIRInit_F32S_F32S`, 1189
- upsampling with filtering —
  - `mlib_SignalUpSampleFIRInit_F32_F32`, 1188
- upsampling with filtering —
  - `mlib_SignalUpSampleFIR_F32S_F32S`, 1184

upsampling with filtering —  
mlib\_SignalUpSampleFIR\_F32\_F32, 1183

## V

vector addition —  
mlib\_VectorAdd\_S16C\_S16C\_Mod, 1207  
vector addition —  
mlib\_VectorAdd\_S16C\_S16C\_Sat, 1207  
vector addition —  
mlib\_VectorAdd\_S16C\_S8C\_Mod, 1207  
vector addition —  
mlib\_VectorAdd\_S16C\_S8C\_Sat, 1207  
vector addition —  
mlib\_VectorAdd\_S16C\_U8C\_Mod, 1207  
vector addition —  
mlib\_VectorAdd\_S16C\_U8C\_Sat, 1207  
vector addition —  
mlib\_VectorAdd\_S16\_S16\_Mod, 1207  
vector addition —  
mlib\_VectorAdd\_S16\_S16\_Sat, 1207  
vector addition —  
mlib\_VectorAdd\_S16\_S8\_Mod, 1207  
vector addition —  
mlib\_VectorAdd\_S16\_S8\_Sat, 1207  
vector addition —  
mlib\_VectorAdd\_S16\_U8\_Mod, 1207  
vector addition —  
mlib\_VectorAdd\_S16\_U8\_Sat, 1207  
vector addition —  
mlib\_VectorAdd\_S32C\_S16C\_Mod, 1207  
vector addition —  
mlib\_VectorAdd\_S32C\_S16C\_Sat, 1207  
vector addition —  
mlib\_VectorAdd\_S32C\_S32C\_Mod, 1207  
vector addition —  
mlib\_VectorAdd\_S32C\_S32C\_Sat, 1207  
vector addition —  
mlib\_VectorAdd\_S32\_S16\_Mod, 1207  
vector addition —  
mlib\_VectorAdd\_S32\_S16\_Sat, 1207  
vector addition —  
mlib\_VectorAdd\_S32\_S32\_Mod, 1207  
vector addition —  
mlib\_VectorAdd\_S32\_S32\_Sat, 1207  
vector addition —  
mlib\_VectorAdd\_S8C\_S8C\_Mod, 1207

vector addition —  
mlib\_VectorAdd\_S8C\_S8C\_Sat, 1207  
vector addition —  
mlib\_VectorAdd\_S8\_S8\_Mod, 1207  
vector addition —  
mlib\_VectorAdd\_S8\_S8\_Sat, 1207  
vector addition —  
mlib\_VectorAdd\_U8C\_U8C\_Mod, 1207  
vector addition —  
mlib\_VectorAdd\_U8C\_U8C\_Sat, 1207  
vector addition —  
mlib\_VectorAdd\_U8\_U8\_Mod, 1207  
vector addition —  
mlib\_VectorAdd\_U8\_U8\_Sat, 1207  
vector addition to scalar —  
mlib\_VectorAddS\_S16C\_S16C\_Mod, 1202  
vector addition to scalar —  
mlib\_VectorAddS\_S16C\_S16C\_Sat, 1202  
vector addition to scalar —  
mlib\_VectorAddS\_S16C\_S8C\_Mod, 1202  
vector addition to scalar —  
mlib\_VectorAddS\_S16C\_S8C\_Sat, 1202  
vector addition to scalar —  
mlib\_VectorAddS\_S16C\_U8C\_Mod, 1202  
vector addition to scalar —  
mlib\_VectorAddS\_S16C\_U8C\_Sat, 1202  
vector addition to scalar —  
mlib\_VectorAddS\_S16\_S16\_Mod, 1202  
vector addition to scalar —  
mlib\_VectorAddS\_S16\_S16\_Sat, 1202  
vector addition to scalar —  
mlib\_VectorAddS\_S16\_S8\_Mod, 1202  
vector addition to scalar —  
mlib\_VectorAddS\_S16\_S8\_Sat, 1202  
vector addition to scalar —  
mlib\_VectorAddS\_S16\_U8\_Mod, 1202  
vector addition to scalar —  
mlib\_VectorAddS\_S16\_U8\_Sat, 1202  
vector addition to scalar —  
mlib\_VectorAddS\_S32C\_S16C\_Mod, 1202  
vector addition to scalar —  
mlib\_VectorAddS\_S32C\_S16C\_Sat, 1202  
vector addition to scalar —  
mlib\_VectorAddS\_S32C\_S32C\_Mod, 1202  
vector addition to scalar —  
mlib\_VectorAddS\_S32C\_S32C\_Sat, 1202  
vector addition to scalar —  
mlib\_VectorAddS\_S32\_S16\_Mod, 1202

vector addition to scalar —  
  mlib\_VectorAddS\_S32\_S16\_Sat, 1202

vector addition to scalar —  
  mlib\_VectorAddS\_S32\_S32\_Mod, 1202

vector addition to scalar —  
  mlib\_VectorAddS\_S32\_S32\_Sat, 1202

vector addition to scalar —  
  mlib\_VectorAddS\_S8C\_S8C\_Mod, 1202

vector addition to scalar —  
  mlib\_VectorAddS\_S8C\_S8C\_Sat, 1202

vector addition to scalar —  
  mlib\_VectorAddS\_S8\_S8\_Mod, 1202

vector addition to scalar —  
  mlib\_VectorAddS\_S8\_S8\_Sat, 1202

vector addition to scalar —  
  mlib\_VectorAddS\_U8C\_U8C\_Mod, 1202

vector addition to scalar —  
  mlib\_VectorAddS\_U8C\_U8C\_Sat, 1202

vector addition to scalar —  
  mlib\_VectorAddS\_U8\_U8\_Mod, 1202

vector addition to scalar —  
  mlib\_VectorAddS\_U8\_U8\_Sat, 1202

vector addition to scalar, in place —  
  mlib\_VectorAddS\_S16C\_Mod, 1200

vector addition to scalar, in place —  
  mlib\_VectorAddS\_S16C\_Sat, 1200

vector addition to scalar, in place —  
  mlib\_VectorAddS\_S16\_Mod, 1200

vector addition to scalar, in place —  
  mlib\_VectorAddS\_S16\_Sat, 1200

vector addition to scalar, in place —  
  mlib\_VectorAddS\_S32C\_Mod, 1200

vector addition to scalar, in place —  
  mlib\_VectorAddS\_S32C\_Sat, 1200

vector addition to scalar, in place —  
  mlib\_VectorAddS\_S32\_Mod, 1200

vector addition to scalar, in place —  
  mlib\_VectorAddS\_S32\_Sat, 1200

vector addition to scalar, in place —  
  mlib\_VectorAddS\_S8C\_Mod, 1200

vector addition to scalar, in place —  
  mlib\_VectorAddS\_S8C\_Sat, 1200

vector addition to scalar, in place —  
  mlib\_VectorAddS\_S8\_Mod, 1200

vector addition to scalar, in place —  
  mlib\_VectorAddS\_S8\_Sat, 1200

vector addition to scalar, in place —  
  mlib\_VectorAddS\_U8C\_Mod, 1200

vector addition to scalar, in place —  
  mlib\_VectorAddS\_U8C\_Sat, 1200

vector addition to scalar, in place —  
  mlib\_VectorAddS\_U8\_Mod, 1200

vector addition to scalar, in place —  
  mlib\_VectorAddS\_U8\_Sat, 1200

vector addition, in place —  
  mlib\_VectorAdd\_S16C\_Mod, 1205

vector addition, in place —  
  mlib\_VectorAdd\_S16C\_Sat, 1205

vector addition, in place —  
  mlib\_VectorAdd\_S16\_Mod, 1205

vector addition, in place —  
  mlib\_VectorAdd\_S16\_Sat, 1205

vector addition, in place —  
  mlib\_VectorAdd\_S32C\_Mod, 1205

vector addition, in place —  
  mlib\_VectorAdd\_S32C\_Sat, 1205

vector addition, in place —  
  mlib\_VectorAdd\_S32\_Mod, 1205

vector addition, in place —  
  mlib\_VectorAdd\_S32\_Sat, 1205

vector addition, in place —  
  mlib\_VectorAdd\_S8C\_Mod, 1205

vector addition, in place —  
  mlib\_VectorAdd\_S8C\_Sat, 1205

vector addition, in place —  
  mlib\_VectorAdd\_S8\_Mod, 1205

vector addition, in place —  
  mlib\_VectorAdd\_S8\_Sat, 1205

vector addition, in place —  
  mlib\_VectorAdd\_U8C\_Mod, 1205

vector addition, in place —  
  mlib\_VectorAdd\_U8C\_Sat, 1205

vector addition, in place —  
  mlib\_VectorAdd\_U8\_Mod, 1205

vector addition, in place —  
  mlib\_VectorAdd\_U8\_Sat, 1205

vector complex magnitude —  
  mlib\_VectorMag\_S16C, 1226

vector complex magnitude —  
  mlib\_VectorMag\_S32C, 1226

vector complex magnitude —  
  mlib\_VectorMag\_S8C, 1226

vector complex magnitude —  
  mlib\_VectorMag\_U8C, 1226

vector complex phase (angle) —  
  mlib\_VectorAng\_S16C, 1210

vector complex phase (angle) —  
  mlib\_VectorAng\_S32C, 1210

vector complex phase (angle) —  
  mlib\_VectorAng\_S8C, 1210

vector complex phase (angle) —  
  mlib\_VectorAng\_U8C, 1210

vector conjugate-symmetric extension —  
  mlib\_VectorConjSymExt\_S16C\_S16C\_Sat, 1214

vector conjugate-symmetric extension —  
  mlib\_VectorConjSymExt\_S32C\_S32C\_Sat, 1214

vector conjugate-symmetric extension —  
  mlib\_VectorConjSymExt\_S8C\_S8C\_Sat, 1214

vector conjugation —  
  mlib\_VectorConj\_S16C\_S16C\_Sat, 1212

vector conjugation —  
  mlib\_VectorConj\_S16C\_Sat, 1213

vector conjugation —  
  mlib\_VectorConj\_S32C\_S32C\_Sat, 1212

vector conjugation —  
  mlib\_VectorConj\_S32C\_Sat, 1213

vector conjugation —  
  mlib\_VectorConj\_S8C\_S8C\_Sat, 1212

vector conjugation —  
  mlib\_VectorConj\_S8C\_Sat, 1213

vector conjugation reversion —  
  mlib\_VectorConjRev\_S16C\_S16C\_Sat, 1211

vector conjugation reversion —  
  mlib\_VectorConjRev\_S32C\_S32C\_Sat, 1211

vector conjugation reversion —  
  mlib\_VectorConjRev\_S8C\_S8C\_Sat, 1211

vector copy — mlib\_VectorCopy\_S16, 1221

vector copy — mlib\_VectorCopy\_S16C, 1221

vector copy — mlib\_VectorCopy\_S32, 1221

vector copy — mlib\_VectorCopy\_S32C, 1221

vector copy — mlib\_VectorCopy\_S8, 1221

vector copy — mlib\_VectorCopy\_S8C, 1221

vector copy — mlib\_VectorCopy\_U8, 1221

vector copy — mlib\_VectorCopy\_U8C, 1221

vector data type convert —  
  mlib\_VectorConvert\_S16C\_S32C\_Mod, 1216

vector data type convert —  
  mlib\_VectorConvert\_S16C\_S32C\_Sat, 1216

vector data type convert —  
  mlib\_VectorConvert\_S16C\_S8C\_Mod, 1216

vector data type convert —  
  mlib\_VectorConvert\_S16C\_S8C\_Sat, 1216

vector data type convert —  
  mlib\_VectorConvert\_S16C\_U8C\_Mod, 1216

vector data type convert —  
  mlib\_VectorConvert\_S16C\_U8C\_Sat, 1216

vector data type convert —  
  mlib\_VectorConvert\_S16\_S32\_Mod, 1216

vector data type convert —  
  mlib\_VectorConvert\_S16\_S32\_Sat, 1216

vector data type convert —  
  mlib\_VectorConvert\_S16\_S8\_Mod, 1216

vector data type convert —  
  mlib\_VectorConvert\_S16\_S8\_Sat, 1216

vector data type convert —  
  mlib\_VectorConvert\_S16\_U8\_Mod, 1216

vector data type convert —  
  mlib\_VectorConvert\_S16\_U8\_Sat, 1216

vector data type convert —  
  mlib\_VectorConvert\_S32C\_S16C\_Mod, 1216

vector data type convert —  
  mlib\_VectorConvert\_S32C\_S16C\_Sat, 1216

vector data type convert —  
  mlib\_VectorConvert\_S32C\_S8C\_Mod, 1216

vector data type convert —  
  mlib\_VectorConvert\_S32C\_S8C\_Sat, 1216

vector data type convert —  
  mlib\_VectorConvert\_S32C\_U8C\_Mod, 1216

vector data type convert —  
  mlib\_VectorConvert\_S32C\_U8C\_Sat, 1216

vector data type convert —  
  mlib\_VectorConvert\_S32\_S16\_Mod, 1216

vector data type convert —  
  mlib\_VectorConvert\_S32\_S16\_Sat, 1216

vector data type convert —  
  mlib\_VectorConvert\_S32\_S8\_Mod, 1216

vector data type convert —  
  mlib\_VectorConvert\_S32\_S8\_Sat, 1216

vector data type convert —  
  mlib\_VectorConvert\_S32\_U8\_Mod, 1216

vector data type convert —  
  mlib\_VectorConvert\_S32\_U8\_Sat, 1216

vector data type convert —  
  mlib\_VectorConvert\_S8C\_S16C\_Mod, 1216

vector data type convert —  
  mlib\_VectorConvert\_S8C\_S16C\_Sat, 1216

vector data type convert —  
  mlib\_VectorConvert\_S8C\_S32C\_Mod, 1216

vector data type convert —  
  mlib\_VectorConvert\_S8C\_S32C\_Sat, 1216

vector data type convert —  
  mlib\_VectorConvert\_S8C\_U8C\_Mod, 1216



vector data type convert —  
 mlib\_VectorConvert\_S8C\_U8C\_Sat, 1216

vector data type convert —  
 mlib\_VectorConvert\_S8\_S16\_Mod, 1216

vector data type convert —  
 mlib\_VectorConvert\_S8\_S16\_Sat, 1216

vector data type convert —  
 mlib\_VectorConvert\_S8\_S32\_Mod, 1216

vector data type convert —  
 mlib\_VectorConvert\_S8\_S32\_Sat, 1216

vector data type convert —  
 mlib\_VectorConvert\_S8\_U8\_Mod, 1216

vector data type convert —  
 mlib\_VectorConvert\_S8\_U8\_Sat, 1216

vector data type convert —  
 mlib\_VectorConvert\_U8C\_S16C\_Mod, 1216

vector data type convert —  
 mlib\_VectorConvert\_U8C\_S16C\_Sat, 1216

vector data type convert —  
 mlib\_VectorConvert\_U8C\_S32C\_Mod, 1216

vector data type convert —  
 mlib\_VectorConvert\_U8C\_S32C\_Sat, 1216

vector data type convert —  
 mlib\_VectorConvert\_U8C\_S8C\_Mod, 1216

vector data type convert —  
 mlib\_VectorConvert\_U8C\_S8C\_Sat, 1216

vector data type convert —  
 mlib\_VectorConvert\_U8\_S16\_Mod, 1216

vector data type convert —  
 mlib\_VectorConvert\_U8\_S16\_Sat, 1216

vector data type convert —  
 mlib\_VectorConvert\_U8\_S32\_Mod, 1216

vector data type convert —  
 mlib\_VectorConvert\_U8\_S32\_Sat, 1216

vector data type convert —  
 mlib\_VectorConvert\_U8\_S8\_Mod, 1216

vector data type convert —  
 mlib\_VectorConvert\_U8\_S8\_Sat, 1216

vector dot product (inner product) —  
 mlib\_VectorDotProd\_S16C\_Sat, 1224

vector dot product (inner product) —  
 mlib\_VectorDotProd\_S16\_Sat, 1224

vector dot product (inner product) —  
 mlib\_VectorDotProd\_S32C\_Sat, 1224

vector dot product (inner product) —  
 mlib\_VectorDotProd\_S32\_Sat, 1224

vector dot product (inner product) —  
 mlib\_VectorDotProd\_S8C\_Sat, 1224

vector dot product (inner product) —  
 mlib\_VectorDotProd\_S8\_Sat, 1224

vector dot product (inner product) —  
 mlib\_VectorDotProd\_U8C\_Sat, 1224

vector dot product (inner product) —  
 mlib\_VectorDotProd\_U8\_Sat, 1224

vector Euclidean distance —  
 mlib\_VectorDistance\_S16\_Sat, 1223

vector Euclidean distance —  
 mlib\_VectorDistance\_S32\_Sat, 1223

vector Euclidean distance —  
 mlib\_VectorDistance\_S8\_Sat, 1223

vector Euclidean distance —  
 mlib\_VectorDistance\_U8\_Sat, 1223

vector linear scaling —  
 mlib\_VectorScale\_S16C\_S16C\_Mod, 1269

vector linear scaling —  
 mlib\_VectorScale\_S16C\_S16C\_Sat, 1269

vector linear scaling —  
 mlib\_VectorScale\_S16C\_S8C\_Mod, 1269

vector linear scaling —  
 mlib\_VectorScale\_S16C\_S8C\_Sat, 1269

vector linear scaling —  
 mlib\_VectorScale\_S16C\_U8C\_Mod, 1269

vector linear scaling —  
 mlib\_VectorScale\_S16C\_U8C\_Sat, 1269

vector linear scaling —  
 mlib\_VectorScale\_S16\_S16\_Mod, 1269

vector linear scaling —  
 mlib\_VectorScale\_S16\_S16\_Sat, 1269

vector linear scaling —  
 mlib\_VectorScale\_S16\_S8\_Mod, 1269

vector linear scaling —  
 mlib\_VectorScale\_S16\_S8\_Sat, 1269

vector linear scaling —  
 mlib\_VectorScale\_S16\_U8\_Mod, 1269

vector linear scaling —  
 mlib\_VectorScale\_S16\_U8\_Sat, 1269

vector linear scaling —  
 mlib\_VectorScale\_S32C\_S16C\_Mod, 1269

vector linear scaling —  
 mlib\_VectorScale\_S32C\_S16C\_Sat, 1269

vector linear scaling —  
 mlib\_VectorScale\_S32C\_S32C\_Mod, 1269

vector linear scaling —  
 mlib\_VectorScale\_S32C\_S32C\_Sat, 1269

vector linear scaling —  
 mlib\_VectorScale\_S32\_S16\_Mod, 1269

vector linear scaling —  
  mlib\_VectorScale\_S32\_S16\_Sat, 1269

vector linear scaling —  
  mlib\_VectorScale\_S32\_S32\_Mod, 1269

vector linear scaling —  
  mlib\_VectorScale\_S32\_S32\_Sat, 1269

vector linear scaling —  
  mlib\_VectorScale\_S8C\_S8C\_Mod, 1269

vector linear scaling —  
  mlib\_VectorScale\_S8C\_S8C\_Sat, 1269

vector linear scaling —  
  mlib\_VectorScale\_S8\_S8\_Mod, 1269

vector linear scaling —  
  mlib\_VectorScale\_S8\_S8\_Sat, 1269

vector linear scaling —  
  mlib\_VectorScale\_U8C\_U8C\_Mod, 1269

vector linear scaling —  
  mlib\_VectorScale\_U8C\_U8C\_Sat, 1269

vector linear scaling —  
  mlib\_VectorScale\_U8\_U8\_Mod, 1269

vector linear scaling —  
  mlib\_VectorScale\_U8\_U8\_Sat, 1269

vector linear scaling, in place —  
  mlib\_VectorScale\_S16C\_Mod, 1267

vector linear scaling, in place —  
  mlib\_VectorScale\_S16C\_Sat, 1267

vector linear scaling, in place —  
  mlib\_VectorScale\_S16\_Mod, 1267

vector linear scaling, in place —  
  mlib\_VectorScale\_S16\_Sat, 1267

vector linear scaling, in place —  
  mlib\_VectorScale\_S32C\_Mod, 1267

vector linear scaling, in place —  
  mlib\_VectorScale\_S32C\_Sat, 1267

vector linear scaling, in place —  
  mlib\_VectorScale\_S32\_Mod, 1267

vector linear scaling, in place —  
  mlib\_VectorScale\_S32\_Sat, 1267

vector linear scaling, in place —  
  mlib\_VectorScale\_S8C\_Mod, 1267

vector linear scaling, in place —  
  mlib\_VectorScale\_S8C\_Sat, 1267

vector linear scaling, in place —  
  mlib\_VectorScale\_S8\_Mod, 1267

vector linear scaling, in place —  
  mlib\_VectorScale\_S8\_Sat, 1267

vector linear scaling, in place —  
  mlib\_VectorScale\_U8C\_Mod, 1267

vector linear scaling, in place —  
  mlib\_VectorScale\_U8C\_Sat, 1267

vector linear scaling, in place —  
  mlib\_VectorScale\_U8\_Mod, 1267

vector linear scaling, in place —  
  mlib\_VectorScale\_U8\_Sat, 1267

vector multiplication —  
  mlib\_VectorMul\_S16C\_S16C\_Mod, 1257

vector multiplication —  
  mlib\_VectorMul\_S16C\_S16C\_Sat, 1257

vector multiplication —  
  mlib\_VectorMul\_S16C\_S8C\_Mod, 1257

vector multiplication —  
  mlib\_VectorMul\_S16C\_S8C\_Sat, 1257

vector multiplication —  
  mlib\_VectorMul\_S16C\_U8C\_Mod, 1257

vector multiplication —  
  mlib\_VectorMul\_S16C\_U8C\_Sat, 1257

vector multiplication —  
  mlib\_VectorMul\_S16\_S16\_Mod, 1257

vector multiplication —  
  mlib\_VectorMul\_S16\_S16\_Sat, 1257

vector multiplication —  
  mlib\_VectorMul\_S16\_S8\_Mod, 1257

vector multiplication —  
  mlib\_VectorMul\_S16\_S8\_Sat, 1257

vector multiplication —  
  mlib\_VectorMul\_S16\_U8\_Mod, 1257

vector multiplication —  
  mlib\_VectorMul\_S16\_U8\_Sat, 1257

vector multiplication —  
  mlib\_VectorMul\_S32C\_S16C\_Mod, 1257

vector multiplication —  
  mlib\_VectorMul\_S32C\_S16C\_Sat, 1257

vector multiplication —  
  mlib\_VectorMul\_S32C\_S32C\_Mod, 1257

vector multiplication —  
  mlib\_VectorMul\_S32C\_S32C\_Sat, 1257

vector multiplication —  
  mlib\_VectorMul\_S32\_S16\_Mod, 1257

vector multiplication —  
  mlib\_VectorMul\_S32\_S16\_Sat, 1257

vector multiplication —  
  mlib\_VectorMul\_S32\_S32\_Mod, 1257

vector multiplication —  
  mlib\_VectorMul\_S32\_S32\_Sat, 1257

vector multiplication —  
  mlib\_VectorMul\_S8C\_S8C\_Mod, 1257



vector multiplication by scalar plus addition —  
 mlib\_VectorMulSAdd\_S32C\_S16C\_Sat, 1239

vector multiplication by scalar plus addition —  
 mlib\_VectorMulSAdd\_S32C\_S32C\_Mod, 1239

vector multiplication by scalar plus addition —  
 mlib\_VectorMulSAdd\_S32C\_S32C\_Sat, 1239

vector multiplication by scalar plus addition —  
 mlib\_VectorMulSAdd\_S32\_S16\_Mod, 1239

vector multiplication by scalar plus addition —  
 mlib\_VectorMulSAdd\_S32\_S16\_Sat, 1239

vector multiplication by scalar plus addition —  
 mlib\_VectorMulSAdd\_S32\_S32\_Mod, 1239

vector multiplication by scalar plus addition —  
 mlib\_VectorMulSAdd\_S32\_S32\_Sat, 1239

vector multiplication by scalar plus addition —  
 mlib\_VectorMulSAdd\_S8C\_S8C\_Mod, 1239

vector multiplication by scalar plus addition —  
 mlib\_VectorMulSAdd\_S8C\_S8C\_Sat, 1239

vector multiplication by scalar plus addition —  
 mlib\_VectorMulSAdd\_S8\_S8\_Mod, 1239

vector multiplication by scalar plus addition —  
 mlib\_VectorMulSAdd\_S8\_S8\_Sat, 1239

vector multiplication by scalar plus addition —  
 mlib\_VectorMulSAdd\_U8C\_U8C\_Mod, 1239

vector multiplication by scalar plus addition —  
 mlib\_VectorMulSAdd\_U8C\_U8C\_Sat, 1239

vector multiplication by scalar plus addition —  
 mlib\_VectorMulSAdd\_U8\_U8\_Mod, 1239

vector multiplication by scalar plus addition —  
 mlib\_VectorMulSAdd\_U8\_U8\_Sat, 1239

vector multiplication by scalar plus addition, in  
 place —  
 mlib\_VectorMulSAdd\_S16C\_Mod, 1237

vector multiplication by scalar plus addition, in  
 place —  
 mlib\_VectorMulSAdd\_S16C\_Sat, 1237

vector multiplication by scalar plus addition, in  
 place —  
 mlib\_VectorMulSAdd\_S16\_Mod, 1237

vector multiplication by scalar plus addition, in  
 place —  
 mlib\_VectorMulSAdd\_S16\_Sat, 1237

vector multiplication by scalar plus addition, in  
 place —  
 mlib\_VectorMulSAdd\_S32C\_Mod, 1237

vector multiplication by scalar plus addition, in  
 place —  
 mlib\_VectorMulSAdd\_S32C\_Sat, 1237

vector multiplication by scalar plus addition, in  
 place —  
 mlib\_VectorMulSAdd\_S32\_Mod, 1237

vector multiplication by scalar plus addition, in  
 place —  
 mlib\_VectorMulSAdd\_S32\_Sat, 1237

vector multiplication by scalar plus addition, in  
 place —  
 mlib\_VectorMulSAdd\_S8C\_Mod, 1237

vector multiplication by scalar plus addition, in  
 place —  
 mlib\_VectorMulSAdd\_S8C\_Sat, 1237

vector multiplication by scalar plus addition, in  
 place —  
 mlib\_VectorMulSAdd\_S8\_Mod, 1237

vector multiplication by scalar plus addition, in  
 place — mlib\_VectorMulSAdd\_S8\_Sat, 1237

vector multiplication by scalar plus addition, in  
 place —  
 mlib\_VectorMulSAdd\_U8C\_Mod, 1237

vector multiplication by scalar plus addition, in  
 place —  
 mlib\_VectorMulSAdd\_U8C\_Sat, 1237

vector multiplication by scalar plus addition, in  
 place —  
 mlib\_VectorMulSAdd\_U8\_Mod, 1237

vector multiplication by scalar plus addition, in  
 place — mlib\_VectorMulSAdd\_U8\_Sat, 1237

vector multiplication by scalar plus shifting —  
 mlib\_VectorMulSShift\_S16C\_S16C\_Mod, 1248

vector multiplication by scalar plus shifting —  
 mlib\_VectorMulSShift\_S16C\_S16C\_Sat, 1248

vector multiplication by scalar plus shifting —  
 mlib\_VectorMulSShift\_S16\_S16\_Mod, 1248

vector multiplication by scalar plus shifting —  
 mlib\_VectorMulSShift\_S16\_S16\_Sat, 1248

vector multiplication by scalar plus shifting —  
 mlib\_VectorMulSShift\_S32C\_S32C\_Mod, 1248

vector multiplication by scalar plus shifting —  
 mlib\_VectorMulSShift\_S32C\_S32C\_Sat, 1248

vector multiplication by scalar plus shifting —  
 mlib\_VectorMulSShift\_S32\_S32\_Mod, 1248

vector multiplication by scalar plus shifting —  
 mlib\_VectorMulSShift\_S32\_S32\_Sat, 1248

vector multiplication by scalar plus shifting —  
 mlib\_VectorMulSShift\_S8C\_S8C\_Mod, 1248

vector multiplication by scalar plus shifting —  
 mlib\_VectorMulSShift\_S8C\_S8C\_Sat, 1248



vector multiplication with shifting —  
 mlib\_VectorMulShift\_S16\_S16\_Mod, 1244

vector multiplication with shifting —  
 mlib\_VectorMulShift\_S16\_S16\_Sat, 1244

vector multiplication with shifting —  
 mlib\_VectorMulShift\_S32C\_S32C\_Mod, 1244

vector multiplication with shifting —  
 mlib\_VectorMulShift\_S32C\_S32C\_Sat, 1244

vector multiplication with shifting —  
 mlib\_VectorMulShift\_S32\_S32\_Mod, 1244

vector multiplication with shifting —  
 mlib\_VectorMulShift\_S32\_S32\_Sat, 1244

vector multiplication with shifting —  
 mlib\_VectorMulShift\_S8C\_S8C\_Mod, 1244

vector multiplication with shifting —  
 mlib\_VectorMulShift\_S8C\_S8C\_Sat, 1244

vector multiplication with shifting —  
 mlib\_VectorMulShift\_S8\_S8\_Mod, 1244

vector multiplication with shifting —  
 mlib\_VectorMulShift\_S8\_S8\_Sat, 1244

vector multiplication with shifting —  
 mlib\_VectorMulShift\_U8C\_U8C\_Mod, 1244

vector multiplication with shifting —  
 mlib\_VectorMulShift\_U8C\_U8C\_Sat, 1244

vector multiplication with shifting —  
 mlib\_VectorMulShift\_U8\_U8\_Mod, 1244

vector multiplication with shifting —  
 mlib\_VectorMulShift\_U8\_U8\_Sat, 1244

vector multiplication with shifting, in place —  
 mlib\_VectorMulShift\_S16C\_Mod, 1242

vector multiplication with shifting, in place —  
 mlib\_VectorMulShift\_S16C\_Sat, 1242

vector multiplication with shifting, in place —  
 mlib\_VectorMulShift\_S16\_Mod, 1242

vector multiplication with shifting, in place —  
 mlib\_VectorMulShift\_S16\_Sat, 1242

vector multiplication with shifting, in place —  
 mlib\_VectorMulShift\_S32C\_Mod, 1242

vector multiplication with shifting, in place —  
 mlib\_VectorMulShift\_S32C\_Sat, 1242

vector multiplication with shifting, in place —  
 mlib\_VectorMulShift\_S32\_Mod, 1242

vector multiplication with shifting, in place —  
 mlib\_VectorMulShift\_S32\_Sat, 1242

vector multiplication with shifting, in place —  
 mlib\_VectorMulShift\_S8C\_Mod, 1242

vector multiplication with shifting, in place —  
 mlib\_VectorMulShift\_S8C\_Sat, 1242

vector multiplication with shifting, in place —  
 mlib\_VectorMulShift\_S8\_Mod, 1242

vector multiplication with shifting, in place —  
 mlib\_VectorMulShift\_S8\_Sat, 1242

vector multiplication with shifting, in place —  
 mlib\_VectorMulShift\_U8C\_Mod, 1242

vector multiplication with shifting, in place —  
 mlib\_VectorMulShift\_U8C\_Sat, 1242

vector multiplication with shifting, in place —  
 mlib\_VectorMulShift\_U8\_Mod, 1242

vector multiplication with shifting, in place —  
 mlib\_VectorMulShift\_U8\_Sat, 1242

vector multiplication, in place —  
 mlib\_VectorMul\_S16C\_Mod, 1255

vector multiplication, in place —  
 mlib\_VectorMul\_S16C\_Sat, 1255

vector multiplication, in place —  
 mlib\_VectorMul\_S16\_Mod, 1255

vector multiplication, in place —  
 mlib\_VectorMul\_S16\_Sat, 1255

vector multiplication, in place —  
 mlib\_VectorMul\_S32C\_Mod, 1255

vector multiplication, in place —  
 mlib\_VectorMul\_S32C\_Sat, 1255

vector multiplication, in place —  
 mlib\_VectorMul\_S32\_Mod, 1255

vector multiplication, in place —  
 mlib\_VectorMul\_S32\_Sat, 1255

vector multiplication, in place —  
 mlib\_VectorMul\_S8C\_Mod, 1255

vector multiplication, in place —  
 mlib\_VectorMul\_S8C\_Sat, 1255

vector multiplication, in place —  
 mlib\_VectorMul\_S8\_Mod, 1255

vector multiplication, in place —  
 mlib\_VectorMul\_S8\_Sat, 1255

vector multiplication, in place —  
 mlib\_VectorMul\_U8C\_Mod, 1255

vector multiplication, in place —  
 mlib\_VectorMul\_U8C\_Sat, 1255

vector multiplication, in place —  
 mlib\_VectorMul\_U8\_Mod, 1255

vector multiplication, in place —  
 mlib\_VectorMul\_U8\_Sat, 1255

vector norm —  
 mlib\_VectorNorm\_S16\_Sat, 1260

vector norm —  
 mlib\_VectorNorm\_S32\_Sat, 1260

vector norm — `mllib_VectorNorm_S8_Sat`, 1260  
 vector norm — `mllib_VectorNorm_U8_Sat`, 1260  
 vector split —  
     `mllib_VectorSplit_S16_S16C`, 1274  
 vector split —  
     `mllib_VectorSplit_S32_S32C`, 1274  
 vector split — `mllib_VectorSplit_S8_S8C`, 1274  
 vector split — `mllib_VectorSplit_U8_U8C`, 1274  
 vector subtraction —  
     `mllib_VectorSub_S16C_S16C_Mod`, 1282  
 vector subtraction —  
     `mllib_VectorSub_S16C_S16C_Sat`, 1282  
 vector subtraction —  
     `mllib_VectorSub_S16C_S8C_Mod`, 1282  
 vector subtraction —  
     `mllib_VectorSub_S16C_S8C_Sat`, 1282  
 vector subtraction —  
     `mllib_VectorSub_S16C_U8C_Mod`, 1282  
 vector subtraction —  
     `mllib_VectorSub_S16C_U8C_Sat`, 1282  
 vector subtraction —  
     `mllib_VectorSub_S16_S16_Mod`, 1282  
 vector subtraction —  
     `mllib_VectorSub_S16_S16_Sat`, 1282  
 vector subtraction —  
     `mllib_VectorSub_S16_S8_Mod`, 1282  
 vector subtraction —  
     `mllib_VectorSub_S16_S8_Sat`, 1282  
 vector subtraction —  
     `mllib_VectorSub_S16_U8_Mod`, 1282  
 vector subtraction —  
     `mllib_VectorSub_S16_U8_Sat`, 1282  
 vector subtraction —  
     `mllib_VectorSub_S32C_S16C_Mod`, 1282  
 vector subtraction —  
     `mllib_VectorSub_S32C_S16C_Sat`, 1282  
 vector subtraction —  
     `mllib_VectorSub_S32C_S32C_Mod`, 1282  
 vector subtraction —  
     `mllib_VectorSub_S32C_S32C_Sat`, 1282  
 vector subtraction —  
     `mllib_VectorSub_S32_S16_Mod`, 1282  
 vector subtraction —  
     `mllib_VectorSub_S32_S16_Sat`, 1282  
 vector subtraction —  
     `mllib_VectorSub_S32_S32_Mod`, 1282  
 vector subtraction —  
     `mllib_VectorSub_S32_S32_Sat`, 1282  
 vector subtraction —  
     `mllib_VectorSub_S8C_S8C_Mod`, 1282  
 vector subtraction —  
     `mllib_VectorSub_S8C_S8C_Sat`, 1282  
 vector subtraction —  
     `mllib_VectorSub_S8_S8_Mod`, 1282  
 vector subtraction —  
     `mllib_VectorSub_S8_S8_Sat`, 1282  
 vector subtraction —  
     `mllib_VectorSub_U8C_U8C_Mod`, 1282  
 vector subtraction —  
     `mllib_VectorSub_U8C_U8C_Sat`, 1282  
 vector subtraction —  
     `mllib_VectorSub_U8_U8_Mod`, 1282  
 vector subtraction —  
     `mllib_VectorSub_U8_U8_Sat`, 1282  
 vector subtraction from scalar —  
     `mllib_VectorSubS_S16C_S16C_Mod`, 1277  
 vector subtraction from scalar —  
     `mllib_VectorSubS_S16C_S16C_Sat`, 1277  
 vector subtraction from scalar —  
     `mllib_VectorSubS_S16C_S8C_Mod`, 1277  
 vector subtraction from scalar —  
     `mllib_VectorSubS_S16C_S8C_Sat`, 1277  
 vector subtraction from scalar —  
     `mllib_VectorSubS_S16C_U8C_Mod`, 1277  
 vector subtraction from scalar —  
     `mllib_VectorSubS_S16C_U8C_Sat`, 1277  
 vector subtraction from scalar —  
     `mllib_VectorSubS_S16_S16_Mod`, 1277  
 vector subtraction from scalar —  
     `mllib_VectorSubS_S16_S16_Sat`, 1277  
 vector subtraction from scalar —  
     `mllib_VectorSubS_S16_S8_Mod`, 1277  
 vector subtraction from scalar —  
     `mllib_VectorSubS_S16_S8_Sat`, 1277  
 vector subtraction from scalar —  
     `mllib_VectorSubS_S16_U8_Mod`, 1277  
 vector subtraction from scalar —  
     `mllib_VectorSubS_S16_U8_Sat`, 1277  
 vector subtraction from scalar —  
     `mllib_VectorSubS_S32C_S16C_Mod`, 1277  
 vector subtraction from scalar —  
     `mllib_VectorSubS_S32C_S16C_Sat`, 1277  
 vector subtraction from scalar —  
     `mllib_VectorSubS_S32C_S32C_Mod`, 1277  
 vector subtraction from scalar —  
     `mllib_VectorSubS_S32C_S32C_Sat`, 1277

vector subtraction from scalar —  
 mlib\_VectorSubS\_S32\_S16\_Mod, 1277

vector subtraction from scalar —  
 mlib\_VectorSubS\_S32\_S16\_Sat, 1277

vector subtraction from scalar —  
 mlib\_VectorSubS\_S32\_S32\_Mod, 1277

vector subtraction from scalar —  
 mlib\_VectorSubS\_S32\_S32\_Sat, 1277

vector subtraction from scalar —  
 mlib\_VectorSubS\_S8C\_S8C\_Mod, 1277

vector subtraction from scalar —  
 mlib\_VectorSubS\_S8C\_S8C\_Sat, 1277

vector subtraction from scalar —  
 mlib\_VectorSubS\_S8\_S8\_Mod, 1277

vector subtraction from scalar —  
 mlib\_VectorSubS\_S8\_S8\_Sat, 1277

vector subtraction from scalar —  
 mlib\_VectorSubS\_U8C\_U8C\_Mod, 1277

vector subtraction from scalar —  
 mlib\_VectorSubS\_U8C\_U8C\_Sat, 1277

vector subtraction from scalar —  
 mlib\_VectorSubS\_U8\_U8\_Mod, 1277

vector subtraction from scalar —  
 mlib\_VectorSubS\_U8\_U8\_Sat, 1277

vector subtraction from scalar, in place —  
 mlib\_VectorSubS\_S16C\_Mod, 1275

vector subtraction from scalar, in place —  
 mlib\_VectorSubS\_S16C\_Sat, 1275

vector subtraction from scalar, in place —  
 mlib\_VectorSubS\_S16\_Mod, 1275

vector subtraction from scalar, in place —  
 mlib\_VectorSubS\_S16\_Sat, 1275

vector subtraction from scalar, in place —  
 mlib\_VectorSubS\_S32C\_Mod, 1275

vector subtraction from scalar, in place —  
 mlib\_VectorSubS\_S32C\_Sat, 1275

vector subtraction from scalar, in place —  
 mlib\_VectorSubS\_S32\_Mod, 1275

vector subtraction from scalar, in place —  
 mlib\_VectorSubS\_S32\_Sat, 1275

vector subtraction from scalar, in place —  
 mlib\_VectorSubS\_S8C\_Mod, 1275

vector subtraction from scalar, in place —  
 mlib\_VectorSubS\_S8C\_Sat, 1275

vector subtraction from scalar, in place —  
 mlib\_VectorSubS\_S8\_Mod, 1275

vector subtraction from scalar, in place —  
 mlib\_VectorSubS\_S8\_Sat, 1275

vector subtraction from scalar, in place —  
 mlib\_VectorSubS\_U8C\_Mod, 1275

vector subtraction from scalar, in place —  
 mlib\_VectorSubS\_U8C\_Sat, 1275

vector subtraction from scalar, in place —  
 mlib\_VectorSubS\_U8\_Mod, 1275

vector subtraction from scalar, in place —  
 mlib\_VectorSubS\_U8\_Sat, 1275

vector subtraction, in place —  
 mlib\_VectorSub\_S16C\_Mod, 1280

vector subtraction, in place —  
 mlib\_VectorSub\_S16C\_Sat, 1280

vector subtraction, in place —  
 mlib\_VectorSub\_S16\_Mod, 1280

vector subtraction, in place —  
 mlib\_VectorSub\_S16\_Sat, 1280

vector subtraction, in place —  
 mlib\_VectorSub\_S32C\_Mod, 1280

vector subtraction, in place —  
 mlib\_VectorSub\_S32C\_Sat, 1280

vector subtraction, in place —  
 mlib\_VectorSub\_S32\_Mod, 1280

vector subtraction, in place —  
 mlib\_VectorSub\_S32\_Sat, 1280

vector subtraction, in place —  
 mlib\_VectorSub\_S8C\_Mod, 1280

vector subtraction, in place —  
 mlib\_VectorSub\_S8C\_Sat, 1280

vector subtraction, in place —  
 mlib\_VectorSub\_S8\_Mod, 1280

vector subtraction, in place —  
 mlib\_VectorSub\_S8\_Sat, 1280

vector subtraction, in place —  
 mlib\_VectorSub\_U8C\_Mod, 1280

vector subtraction, in place —  
 mlib\_VectorSub\_U8C\_Sat, 1280

vector subtraction, in place —  
 mlib\_VectorSub\_U8\_Mod, 1280

vector subtraction, in place —  
 mlib\_VectorSub\_U8\_Sat, 1280

vector merge —  
 mlib\_VectorMerge\_S16C\_S16, 1229

vector merge —  
 mlib\_VectorMerge\_S32C\_S32, 1229

vector merge —  
 mlib\_VectorMerge\_S8C\_S8, 1229

vector merge —  
 mlib\_VectorMerge\_U8C\_U8, 1229



## W

wavelet transformation —  
    mllib\_VideoWaveletForwardTwoTenTrans, 1536

wavelet transformation —  
    mllib\_VideoWaveletForwardTwoTenTrans\_ S16\_S16, 1536

wavelet transformation —  
    mllib\_VideoWaveletForwardTwoTenTrans\_ S16\_U8, 1536

wavelet transformation —  
    mllib\_VideoWaveletForwardTwoTenTrans\_ S32\_S16, 1536

wavelet transformation —  
    mllib\_VideoWaveletForwardTwoTenTrans\_ S32\_S32, 1536

wavelet transformation —  
    mllib\_VideoWaveletInverseTwoTenTrans, 1537

wavelet transformation —  
    mllib\_VideoWaveletInverseTwoTenTrans\_ S16\_S16, 1537

wavelet transformation —  
    mllib\_VideoWaveletInverseTwoTenTrans\_ S16\_S32, 1537

wavelet transformation —  
    mllib\_VideoWaveletInverseTwoTenTrans\_ S32\_S32, 1537

wavelet transformation —  
    mllib\_VideoWaveletInverseTwoTenTrans\_ U8\_S16, 1537

wavelet transformation, sign-magnitude conversion —  
    mllib\_VideoSignMagnitudeConvert\_S16, 1523

wavelet transformation, sign-magnitude conversion —  
    mllib\_VideoSignMagnitudeConvert\_S16\_S16, 1524

wavelet transformation, sign-magnitude conversion —  
    mllib\_VideoSignMagnitudeConvert\_S32, 1525

wavelet transformation, sign-magnitude conversion —  
    mllib\_VideoSignMagnitudeConvert\_S32\_S32, 1526

white noise generation —  
    mllib\_SignalWhiteNoiseFree\_F32, 1195

white noise generation —  
    mllib\_SignalWhiteNoiseFree\_S16, 1196

white noise generation —  
    mllib\_SignalWhiteNoiseInit\_F32, 1197

white noise generation —  
    mllib\_SignalWhiteNoiseInit\_S16, 1198

white noise generation —  
    mllib\_SignalWhiteNoise\_F32, 1194

white noise generation —  
    mllib\_SignalWhiteNoise\_S16, 1199

window-level operation —  
    mllib\_VolumeWindowLevel, 1545

windowing —  
    mllib\_SignalMulWindow\_F32, 1152

windowing —  
    mllib\_SignalMulWindow\_F32S, 1154

windowing —  
    mllib\_SignalMulWindow\_F32S\_F32S, 1155

windowing —  
    mllib\_SignalMulWindow\_F32\_F32, 1153

windowing —  
    mllib\_SignalMulWindow\_S16, 1156

windowing —  
    mllib\_SignalMulWindow\_S16S, 1156

windowing —  
    mllib\_SignalMulWindow\_S16S\_S16S, 1157

windowing —  
    mllib\_SignalMulWindow\_S16\_S16, 1157

## X

X-axis flip — mllib\_ImageFlipX, 465

X-axis flip — mllib\_ImageFlipX\_Fp, 466

Xor — mllib\_ImageXor, 751

Xor, in place — mllib\_ImageXor\_Inp, 752

Xor with a constant —  
    mllib\_ImageConstXor, 369

Xor with a constant, in place —  
    mllib\_ImageConstXor\_Inp, 370

XYZ to RGB color conversion —  
    mllib\_ImageColorXYZ2RGB, 327

XYZ to RGB color conversion —  
    mllib\_ImageColorXYZ2RGB\_Fp, 328

## Y

Y-axis flip — mllib\_ImageFlipY, 467

Y-axis flip — mllib\_ImageFlipY\_Fp, 468

YCC to RGB color conversion —  
    mllib\_ImageColorYCC2RGB, 329

- YCC to RGB color conversion —
  - mllib\_ImageColorYCC2RGB\_Fp, 330
- YUV to RGB color conversion —
  - mllib\_VideoColorYUV2ABGR411, 1357
- YUV to RGB color conversion —
  - mllib\_VideoColorYUV2ABGR420, 1359
- YUV to RGB color conversion —
  - mllib\_VideoColorYUV2ABGR420\_W, 1361
- YUV to RGB color conversion —
  - mllib\_VideoColorYUV2ABGR420\_WX2, 1363
- YUV to RGB color conversion —
  - mllib\_VideoColorYUV2ABGR420\_WX3, 1365
- YUV to RGB color conversion —
  - mllib\_VideoColorYUV2ABGR420\_X2, 1367
- YUV to RGB color conversion —
  - mllib\_VideoColorYUV2ABGR420\_X3, 1369
- YUV to RGB color conversion —
  - mllib\_VideoColorYUV2ABGR422, 1371
- YUV to RGB color conversion —
  - mllib\_VideoColorYUV2ABGR444, 1373
- YUV to RGB color conversion —
  - mllib\_VideoColorYUV2ARGB411, 1375
- YUV to RGB color conversion —
  - mllib\_VideoColorYUV2ARGB420, 1377
- YUV to RGB color conversion —
  - mllib\_VideoColorYUV2ARGB422, 1379
- YUV to RGB color conversion —
  - mllib\_VideoColorYUV2ARGB444, 1381
- YUV to RGB color conversion —
  - mllib\_VideoColorYUV2RGB411, 1383
- YUV to RGB color conversion —
  - mllib\_VideoColorYUV2RGB420, 1385
- YUV to RGB color conversion —
  - mllib\_VideoColorYUV2RGB422, 1387
- YUV to RGB color conversion —
  - mllib\_VideoColorYUV2RGB444, 1389
- zoom, with translation, with table-driven interpolation —
  - mllib\_ImageZoomTranslateTable, 782
- zoom, with translation, with table-driven interpolation —
  - mllib\_ImageZoomTranslateTable\_Fp, 787
- zoom on color-indexed image —
  - mllib\_ImageZoomIndex, 768

## Z

- zoom — mllib\_ImageZoom, 757
- zoom — mllib\_ImageZoom\_Fp, 761
- zoom, with translation —
  - mllib\_ImageZoomTranslate, 775
- zoom, with translation —
  - mllib\_ImageZoomTranslate\_Fp, 780
- zoom, with translation, and convert to grayscale
  - mllib\_ImageZoomTranslateToGray, 789