

MBS Real Studio Picture Plugin Documentation

Christian Schmitz

May 15, 2012

0.1 Introduction

This is the PDF version of the documentation for the Real Studio Plug-in from Monkeybread Software Germany. Plugin part: MBS Real Studio Picture Plugin

0.2 Content

- 1 List of all topics 3
- 2 All items in this plugin 15
- 8 List of all classes 173
- 9 List of all global methods 175

Chapter 1

List of Topics

• 2 Graphics & Pictures	15
– 2.1 Globals	15
* 2.1 BlendPicturesMBS(result as picture, source as picture, sourcepercent as double, dest as picture, destpercent as double, x As Integer, y As Integer, width As Integer, height As Integer) as boolean	15
* 2.1 BlendPicturesMBS(source as picture, sourcepercent as double, dest as picture, destpercent as double) as picture	16
* 2.1 BlendPicturesWithMaskMBS(result as picture, source as picture, dest as picture, mask as picture, x As Integer, y As Integer, width As Integer, height As Integer) as boolean	16
* 2.1 BlendPicturesWithMaskMBS(source as picture, dest as picture, mask as picture) as picture	17
* 2.1 BlendPicturesWithMaskWithBackgroundMBS(SourceImage As Picture, DestImage As Picture, Mask As Picture, Result As Picture, X As Integer, Y As Integer, Width As Integer, Height As Integer) as boolean	18
* 2.1 BlendPicturesWithMaskWithBackgroundMBS(SourceImage As Picture, DestImage As Picture, Mask As Picture, Result As Picture, X As Integer, Y As Integer, Width As Integer, Height As Integer, BackgroundColour As Color) as boolean	18
* 2.1 ColorizePictureMBS(Pict As Picture, Mask As Picture, foreR as double, foreG as double, foreB as double, foreA as double, backR as double, backG as double, backB as double, backA as double) as boolean	19
* 2.1 CombinePicturesMBS(red as picture, blue as picture, green as picture) as picture	19
* 2.1 DiffPicturesMBS(source as picture, dest as picture, square as boolean) as picture	19
* 2.1 GetMBfromPictureMBS(pic as picture, mask as picture, mode as string) as memoryblock	20
* 2.1 GetMBfromPictureMBS(pic as picture, mode as string) as memoryblock	21
* 2.1 MandelbrotSetMBS(Threaded as integer, width as integer, height as integer, fx as double = 4.0, fy as double = 4.0, dx as double = -2.0, dy as double = -2.0) as picture	21

* 2.1 MemoryblockABGRtoPictureMBS(dest as picture, source as memoryblock, offset as integer, width as integer, height as integer) as picture	22
* 2.1 MemoryblockABGRtoPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer) as picture	22
* 2.1 MemoryblockARGBtoPictureMBS(dest as picture, source as memoryblock, offset as integer, width as integer, height as integer) as picture	23
* 2.1 MemoryblockARGBtoPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer) as picture	24
* 2.1 MemoryblockARGBtoPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer, LittleEndian as boolean) as picture	24
* 2.1 MemoryblockBGRtoPictureMBS(dest as picture, source as memoryblock, offset as integer, width as integer, height as integer) as picture	26
* 2.1 MemoryblockBGRtoPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer) as picture	26
* 2.1 MemoryblockBGRtoPictureMBS(dest as picture, source as memoryblock, offset as integer, width as integer, height as integer) as picture	27
* 2.1 MemoryblockBGRtoPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer) as picture	28
* 2.1 MemoryblockGrayToPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer, PixelByteSize as integer) as picture	29
* 2.1 MemoryblockGrayToPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer, PixelByteSize as integer, Red as integer, Blue as integer, Green as integer) as picture	30
* 2.1 MemoryblockGrayToPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer, PixelByteSize as integer, Red() as integer, Blue() as integer, Green() as integer) as picture	31
* 2.1 MemoryblockRGBAtoPictureMBS(dest as picture, source as memoryblock, offset as integer, width as integer, height as integer, FlipVertically as boolean=false) as picture	31
* 2.1 MemoryblockRGBAtoPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer, FlipVertically as boolean=false) as picture	32
* 2.1 MemoryblockRGBtoPictureMBS(dest as picture, source as memoryblock, offset as integer, width as integer, height as integer) as picture	33
* 2.1 MemoryblockRGBtoPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer) as picture	34
* 2.1 NewBluePaletteMBS as PaletteMBS	35
* 2.1 NewGrayPaletteMBS as PaletteMBS	36
* 2.1 NewGreenPaletteMBS as PaletteMBS	36
* 2.1 NewPaletteMBS(count as integer) as PaletteMBS	36
* 2.1 NewPalmPaletteMBS as PaletteMBS	37
* 2.1 NewPictureEditorMBS(pic as picture) as PictureEditorMBS	37
* 2.1 NewPictureMBS(width as integer, height as integer, pixeltype as integer, buffer as memoryblock, rowbytes as integer) as picture	38
* 2.1 NewPictureReaderMBS(pic as picture) as PictureReaderMBS	38
* 2.1 NewPictureWithColorMBS(width as integer, height as integer, c as color) as picture	40

* 2.1 NewPictureWriterMBS(width as integer, height as integer) as PictureWriterMBS	40
* 2.1 NewRedPaletteMBS as PaletteMBS	41
* 2.1 NewSystemPaletteMBS as PaletteMBS	42
* 2.1 NewWebPaletteMBS as PaletteMBS	42
* 2.1 NewWindowsPaletteMBS as PaletteMBS	43
* 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean) as boolean	43
* 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color) as boolean	45
* 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color, MaskColour As color) as boolean	48
* 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer) as boolean	49
* 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer, MaskColour As Integer) as boolean	51
* 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean) as boolean	54
* 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color) as boolean	56
* 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color, MaskColour As color) as boolean	58
* 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer) as boolean	60
* 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer, MaskColour As Integer) as boolean	62
* 2.1 PictureCopyPixelFastMBS(DestImage As Picture, Source As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer) as boolean	63
* 2.1 PtrABGRtoPictureMBS(dest as picture, source as Ptr, offset as integer, width as integer, height as integer) as picture	65

* 2.1 PtrABGRtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer) as picture	65
* 2.1 PtrARGBtoPictureMBS(dest as picture, source as Ptr, offset as integer, width as integer, height as integer) as picture	66
* 2.1 PtrARGBtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer) as picture	66
* 2.1 PtrARGBtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer, LittleEndian as boolean) as picture	67
* 2.1 PtrBGRtoPictureMBS(dest as picture, source as Ptr, offset as integer, width as integer, height as integer) as picture	68
* 2.1 PtrBGRtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer) as picture	68
* 2.1 PtrBGRtoPictureMBS(dest as picture, source as Ptr, offset as integer, width as integer, height as integer) as picture	69
* 2.1 PtrBGRtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer) as picture	69
* 2.1 PtrGrayToPictureMBS(source as Ptr, offset as integer, width as integer, height as integer, PixelByteSize as integer) as picture	70
* 2.1 PtrGrayToPictureMBS(source as Ptr, offset as integer, width as integer, height as integer, PixelByteSize as integer, Red as integer, Blue as integer, Green as integer) as picture	70
* 2.1 PtrGrayToPictureMBS(source as Ptr, offset as integer, width as integer, height as integer, PixelByteSize as integer, Red() as integer, Blue() as integer, Green() as integer) as picture	71
* 2.1 PtrRGBtoPictureMBS(dest as picture, source as Ptr, offset as integer, width as integer, height as integer, FlipVertically as boolean=false) as picture	72
* 2.1 PtrRGBtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer, FlipVertically as boolean=false) as picture	73
* 2.1 PtrRGBtoPictureMBS(dest as picture, source as Ptr, offset as integer, width as integer, height as integer) as picture	73
* 2.1 PtrRGBtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer) as picture	74
* 2.1 TintPictureMBS(source as picture, GreyBase as color, SepiaBase as color) as picture	74
– 2.2 class PictureConvolutionMBS	76
* 2.2.1 close	77
* 2.2.1 Hor(index as integer) as double	77
* 2.2.1 Run(channels as integer) as boolean	77
* 2.2.1 Ver(index as integer) as double	79
* 2.2.2 DestinationPicture as Picture	79
* 2.2.2 SourcePicture as Picture	80
* 2.2.2 ValueCount as Integer	80
– 2.3 class PictureEditorMBS	81
* 2.3.1 Data(Row as integer) as MemoryBlock	81
* 2.3.2 AllData as Memoryblock	81
* 2.3.2 AllDataCopy as Memoryblock	82

* 2.3.2 BlueOffset as Integer	82
* 2.3.2 BytesPerPixel as Integer	82
* 2.3.2 DataPtr as Integer	82
* 2.3.2 GreenOffset as Integer	83
* 2.3.2 Height as Integer	83
* 2.3.2 Picture as Picture	83
* 2.3.2 RedOffset as Integer	83
* 2.3.2 RowBytes as Integer	83
* 2.3.2 Width as Integer	84
– 2.12 class PaletteMBS	120
* 2.12.1 Col(i as integer) as color	120
* 2.12.1 CountColors as integer	120
* 2.12.1 Mem as memoryblock	120
* 2.12.1 NewPicture(width as integer,height as integer) as picture	121
* 2.12.2 count as integer	121
– 2.10 class PictureMinMaxMBS	107
* 2.10.1 FindAll(p as picture) as boolean	107
* 2.10.1 FindBlue(p as picture) as boolean	107
* 2.10.1 FindGreen(p as picture) as boolean	108
* 2.10.1 FindMaxAll(p as picture) as boolean	108
* 2.10.1 FindMaxBlue(p as picture) as boolean	108
* 2.10.1 FindMaxGreen(p as picture) as boolean	108
* 2.10.1 FindMaxRed(p as picture) as boolean	108
* 2.10.1 FindMaxSum(p as picture) as boolean	109
* 2.10.1 FindMinAll(p as picture) as boolean	109
* 2.10.1 FindMinBlue(p as picture) as boolean	109
* 2.10.1 FindMinGreen(p as picture) as boolean	109
* 2.10.1 FindMinRed(p as picture) as boolean	109
* 2.10.1 FindMinSum(p as picture) as boolean	110
* 2.10.1 FindRed(p as picture) as boolean	110
* 2.10.1 FindSum(p as picture) as boolean	110
* 2.10.2 BlueMax as Integer	110
* 2.10.2 BlueMaxX as Integer	111
* 2.10.2 BlueMaxY as Integer	111
* 2.10.2 BlueMin as Integer	111
* 2.10.2 BlueMinX as Integer	111
* 2.10.2 BlueMinY as Integer	112
* 2.10.2 GreenMax as Integer	112
* 2.10.2 GreenMaxX as Integer	112
* 2.10.2 GreenMaxY as Integer	112
* 2.10.2 GreenMin as Integer	113

* 2.10.2 GreenMinX as Integer	113
* 2.10.2 GreenMinY as Integer	113
* 2.10.2 RedMax as Integer	113
* 2.10.2 RedMaxX as Integer	114
* 2.10.2 RedMaxY as Integer	114
* 2.10.2 RedMin as Integer	114
* 2.10.2 RedMinX as Integer	114
* 2.10.2 RedMinY as Integer	115
* 2.10.2 SumMax as Integer	115
* 2.10.2 SumMaxX as Integer	115
* 2.10.2 SumMaxY as Integer	115
* 2.10.2 SumMin as Integer	116
* 2.10.2 SumMinX as Integer	116
* 2.10.2 SumMinY as Integer	116
– 2.4 class PictureReaderMBS	84
* 2.4.1 Data(Row as integer) as MemoryBlock	86
* 2.4.2 BlueOffset as Integer	86
* 2.4.2 BytesPerPixel as Integer	86
* 2.4.2 Data as Memoryblock	87
* 2.4.2 DataCopy as Memoryblock	87
* 2.4.2 DataPtr as Integer	87
* 2.4.2 GreenOffset as Integer	87
* 2.4.2 Height as Integer	88
* 2.4.2 Picture as Picture	88
* 2.4.2 RedOffset as Integer	88
* 2.4.2 RowBytes as Integer	88
* 2.4.2 Width as Integer	88
– 2.5 class PictureWriterMBS	89
* 2.5.1 Data(Row as integer) as MemoryBlock	90
* 2.5.1 Render as picture	90
* 2.5.2 BlueOffset as Integer	91
* 2.5.2 BytesPerPixel as Integer	91
* 2.5.2 Data as Memoryblock	91
* 2.5.2 DataCopy as Memoryblock	92
* 2.5.2 DataPtr as Integer	92
* 2.5.2 GreenOffset as Integer	92
* 2.5.2 Height as Integer	92
* 2.5.2 Picture as Picture	93
* 2.5.2 RedOffset as Integer	93
* 2.5.2 RowBytes as Integer	93
* 2.5.2 Width as Integer	93

– 2.6 class PictureSepiaMBS	94
* 2.6.1 close	94
* 2.6.1 Run as boolean	94
* 2.6.2 DestinationPicture as Picture	95
* 2.6.2 FactorBlue as Double	95
* 2.6.2 FactorGreen as Double	95
* 2.6.2 FactorRed as Double	96
* 2.6.2 MaxX as Integer	96
* 2.6.2 MaxY as Integer	96
* 2.6.2 MinX as Integer	96
* 2.6.2 MinY as Integer	97
* 2.6.2 SepiaBlue as Integer	97
* 2.6.2 SepiaGreen as Integer	97
* 2.6.2 SepiaRed as Integer	97
* 2.6.2 SourcePicture as Picture	98
– 2.7 class PictureMatrixMBS	98
* 2.7.1 close	98
* 2.7.1 Matrix(x as integer, y as integer) as integer	98
* 2.7.1 Run as boolean	99
* 2.7.1 RunRGB(red as boolean, green as boolean, blue as boolean) as boolean	99
* 2.7.2 DestinationPicture as Picture	99
* 2.7.2 Displacement as Integer	99
* 2.7.2 MaxX as Integer	100
* 2.7.2 MaxY as Integer	100
* 2.7.2 MinX as Integer	100
* 2.7.2 MinY as Integer	100
* 2.7.2 ScaleFactor as Double	101
* 2.7.2 SourcePicture as Picture	101
– 2.9 class PictureMatrix3DMBS	104
* 2.9.1 close	104
* 2.9.1 Matrix(x as integer, y as integer) as double	104
* 2.9.1 Run as boolean	105
* 2.9.2 DestinationPicture as Picture	105
* 2.9.2 MaxX as Integer	105
* 2.9.2 MaxY as Integer	105
* 2.9.2 MinX as Integer	106
* 2.9.2 MinY as Integer	106
* 2.9.2 SourcePicture as Picture	106
– 2.8 class PictureLut3DMBS	101
* 2.8.1 close	101
* 2.8.1 Run as boolean	102

* 2.8.1 Table(r as integer, g as integer, b as integer, x as integer) as double	102
* 2.8.2 DestinationPicture as Picture	102
* 2.8.2 MaxX as Integer	103
* 2.8.2 MaxY as Integer	103
* 2.8.2 MinX as Integer	103
* 2.8.2 MinY as Integer	103
* 2.8.2 SourcePicture as Picture	104
– 2.11 class BarcodeScannerMBS	116
* 2.11.1 Scan(p as picture) as boolean	117
* 2.11.1 Scan(p as picture, lines() as integer) as boolean	117
* 2.11.2 Barcode as String	118
* 2.11.2 CheckDigits as Boolean	118
* 2.11.2 LastBarcode as String	119
* 2.11.2 LastPicture as Picture	119
* 2.11.2 MinimumLength as Integer	119
* 2.11.2 Mode as Integer	119
– 2.13 class PaletteCalculatorMBS	121
* 2.13.1 Col(i as integer) as color	122
* 2.13.1 CountColors as integer	122
* 2.13.1 CreatePicturePalette(Pic as picture) as integer	122
* 2.13.1 GetIndexOfColor(col as color) as integer	122
* 2.13.1 GetIndexOfColor(r as integer, g as integer, b as integer) as integer	123
* 2.13.1 GetNearestIndexOfColor(col as color) as integer	123
* 2.13.1 GetNearestIndexOfColor(r as integer, g as integer, b as integer) as integer	123
* 2.13.1 Transform(mem as memoryblock, width as integer, height as integer) as picture	124
* 2.13.1 Transform(Pic as picture) as memoryblock	124
* 2.13.1 TransformBetterDithering(Pic as picture) as memoryblock	124
* 2.13.1 TransformFastDithering(Pic as picture) as memoryblock	125
* 2.13.2 Count as Integer	125
• 3 Icon Service	127
– 3.3 class IconMBS	143
* 3.3.1 Constructor(f as folderitem)	145
* 3.3.1 Constructor(type as string, creator as string)	145
* 3.3.1 Constructor(type as string, creator as string, extension as string, mime as string)	146
* 3.3.1 DrawIcon(g as graphics,x as integer,y as integer,width as integer,height as integer)	147
* 3.3.1 DrawIcon(g as graphics,x as integer,y as integer,width as integer,height as integer,align as integer)	148
* 3.3.1 DrawIcon(g as graphics,x as integer,y as integer,width as integer,height as integer,align as integer,transform as integer)	148

* 3.3.1 DrawIconCGContext(CGContextHandle as integer,x as integer,y as integer,width as integer,height as integer, align as integer, transform as integer, flags as integer, labelColor as color)	150
* 3.3.1 GetBackground as IconMBS	151
* 3.3.1 GetForeground as IconMBS	152
* 3.3.1 IconFamily as IconFamilyMBS	152
* 3.3.1 IsIconRefMaskEmpty as boolean	153
* 3.3.1 PointInIcon(pointx as integer,pointy as integer,x as integer,y as integer,width as integer,height as integer,align as integer) as boolean	153
* 3.3.1 RectInIcon(rectx as integer,recty as integer,rectwidth as integer,rectheight as integer,x as integer,y as integer,width as integer,height as integer,align as integer) as boolean	153
* 3.3.1 RetainCount as integer	154
* 3.3.2 handle as integer	154
* 3.3.2 LastError as integer	155
* 3.3.2 Release as boolean	156
* 3.3.2 valid as boolean	156
– 3.1 Globals	127
* 3.1 CompositeIconsMBS(ForeGround as IconMBS, BackGround as IconMBS) as IconMBS	127
* 3.1 NewIconFamilyMBS as IconFamilyMBS	128
* 3.1 NewIconFamilyMBSFromScrap as IconFamilyMBS	128
– 3.2 class IconFamilyMBS	128
* 3.2.1 close	129
* 3.2.1 Data as string	129
* 3.2.1 GetIconImage(size as integer, byref pic as picture, byref mask as picture) as boolean	130
* 3.2.1 Huge1BitData as picture	130
* 3.2.1 Huge1BitMask as picture	131
* 3.2.1 Huge32BitData as picture	131
* 3.2.1 Huge4BitData as picture	132
* 3.2.1 Huge8BitData as picture	132
* 3.2.1 Huge8BitMask as picture	132
* 3.2.1 Large1BitData as picture	133
* 3.2.1 Large1BitMask as picture	133
* 3.2.1 Large32BitData as picture	134
* 3.2.1 Large4BitData as picture	134
* 3.2.1 Large8BitData as picture	134
* 3.2.1 Large8BitMask as picture	135
* 3.2.1 PutOnScrap	135
* 3.2.1 Register(creator as string, type as string) as IconMBS	136
* 3.2.1 SetIconImage(pic as picture, mask as picture) as boolean	136
* 3.2.1 Small1BitData as picture	137

* 3.2.1 Small1BitMask as picture	137
* 3.2.1 Small32BitData as picture	137
* 3.2.1 Small4BitData as picture	138
* 3.2.1 Small8BitData as picture	138
* 3.2.1 Small8BitMask as picture	139
* 3.2.1 Thumbnail32BitData as picture	139
* 3.2.1 Thumbnail8BitMask as picture	141
* 3.2.1 WriteFile(f as folderitem)	141
* 3.2.2 Dither as boolean	142
* 3.2.2 Handle as integer	142
* 3.2.2 GetLastError as integer	142
* 3.2.2 Release as boolean	143
* 3.2.2 Valid as boolean	143
• 4 Mac	157
– 4.1 Globals	157
* 4.1 SetDesktopPictureMBS(file as folderitem) as integer	157
• 5 Pictures Import and Export	159
– 5.1 Globals	159
* 5.1 BinaryStringtoPictureMBS(data as String) as Picture	159
* 5.1 BMPStringtoPictureMBS(data as string) as picture	160
* 5.1 MergePictureMBS(source1 as picture, source2 as picture) as picture	160
* 5.1 PicturetoBinaryStringMBS(p as picture) as string	161
* 5.1 RenderSamplesMBS(Samples as memoryblock, SampleCount as integer, Smooth as integer, Width as integer, Height as integer, outlinewidth as integer, BackColor as color=& c88B5C4, ForeColor as color=& c274C5A, OutLineColor as color=& c203F4E) as Picture	162
• 6 Screenshot	165
– 6.1 Globals	165
* 6.1 ScreenshotDisplayMBS(index as integer) as picture	165
* 6.1 ScreenshotFromStringMBS(Width as integer, Height as integer, RowBytes as integer, data as string) as picture	166
* 6.1 ScreenshotMBS as picture	166
* 6.1 ScreenshotRectMBS(left as integer, top as integer, width as integer, height as integer) as picture	167
* 6.1 ScreenshotRectMBS(left as integer, top as integer, width as integer, height as integer, destwidth as integer, destheight as integer) as picture	167
* 6.1 ScreenshotStringDisplayMBS(byref Width as integer, byref Height as integer, byref RowBytes as integer, index as integer) as string	168
* 6.1 ScreenshotStringMBS(byref Width as integer, byref Height as integer, byref RowBytes as integer) as string	168

• 7 X-Face	171
– 7.1 Globals	171
* 7.1 PictureFromXFaceMemoryBlockMBS(xface as memoryblock) as picture	171
* 7.1 PictureFromXFaceMemoryBlockMBS(xface as memoryblock, size as integer) as picture	171
* 7.1 PictureFromXFaceStringMBS(xface as string) as picture	172
* 7.1 XFaceStringFromPictureMBS(pic as picture) as string	172

Chapter 2

Graphics & Pictures

2.1 Globals

BlendPicturesMBS(result as picture, source as picture, sourcepercent as double, dest as picture, destpercent as double, x As Integer, y As Integer, width As Integer, height As Integer) as boolean

Plugin Version: 8.7 Console & Web: No Mac: Yes, Win: Yes, Linux: No, . **Function:** Blends two pictures.

Example:

dim a,b,c as picture

a=newpicture(100,100,32)

b=newpicture(100,100,32)

' ... draw something in a and b

c=newpicture(100,100,32)

call BlendPicturesMBS(c, a,0.5,b,0.5, 0, 0, 100, 100)

Notes:

Percent is in range from 0 to 1. Values out of this range may work, but you get strange results.

Reason for returning false:

- One of the pictures used is nil.

- The result picture must be a 24 bit or a 32 bit picture.
- The two parameter pictures have not the same size as the others.

BlendPicturesMBS(source as picture, sourcepercent as double, dest as picture, destpercent as double) as picture

Plugin Version: 4.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Blends two pictures.

Example:

```
dim a,b,c as picture
```

```
a=newpicture(100,100,32)
b=newpicture(100,100,32)
' ... draw something in a and b
c=newpicture(100,100,32)
c=BlendPicturesMBS(a,0.5,b,0.5)
```

Notes:

Percent is in range from 0 to 1. Values out of this range may work, but you get strange results.

Reason for returning nil:

- One of the two pictures used is nil.
- One of the pictures is not a 32bit bitmap picture.
- The two parameter pictures have not the same size as the others.

BlendPicturesWithMaskMBS(result as picture, source as picture, dest as picture, mask as picture, x As Integer, y As Integer, width As Integer, height As Integer) as boolean

Plugin Version: 8.7 Console & Web: No Mac: Yes, Win: Yes, Linux: No, . **Function:** Blends two pictures.

Example:

```
dim a,b,c,m as picture
```

```
a=newpicture(100,100,32)
b=newpicture(100,100,32)
m=newpicture(100,100,32)
' ... draw something in a and b
```

`call BlendPicturesWithMaskMBS(c,a,b,m,0,0,a.width,a.height)`

Notes:

The mask defines how much from one picture is used.

Reason for returning false:

- One of the pictures used is nil.
- The result picture must be a 24 bit or a 32 bit picture.
- The three parameter pictures have not the same size as the others.

BlendPicturesWithMaskMBS(source as picture, dest as picture, mask as picture) as picture

Plugin Version: 4.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Blends two pictures.

Example:

`dim a,b,c,m as picture`

```
a=newpicture(100,100,32)
b=newpicture(100,100,32)
m=newpicture(100,100,32)
' ... draw something in a and b
c=BlendPicturesWithMaskMBS(a,b,m)
```

Notes:

The mask defines how much from one picture is used.

Reason for returning false:

- One of the three pictures used is nil.
- One of the pictures is not a 32bit bitmap picture.
- The three parameter pictures have not the same size as the others.

BlendPicturesWithMaskWithBackgroundMBS(SourceImage As Picture, DestImage As Picture, Mask As Picture, Result As Picture, X As Integer, Y As Integer, Width As Integer, Height As Integer) as boolean

Plugin Version: 8.7 Console & Web: No Mac: Yes, Win: Yes, Linux: No, . **Function:** Blends a picture with another picture.

Notes:

If DestImage is nil, white is used for the background.
If no mask is specified, a full black mask is used.

Result must be a 24bit or 32bit picture.
See also:

- 2.1 BlendPicturesWithMaskWithBackgroundMBS(SourceImage As Picture, DestImage As Picture, Mask As Picture, Result As Picture, X As Integer, Y As Integer, Width As Integer, Height As Integer, BackgroundColour As Color) as boolean 18

BlendPicturesWithMaskWithBackgroundMBS(SourceImage As Picture, DestImage As Picture, Mask As Picture, Result As Picture, X As Integer, Y As Integer, Width As Integer, Height As Integer, BackgroundColour As Color) as boolean

Plugin Version: 8.7 Console & Web: No Mac: Yes, Win: Yes, Linux: No, . **Function:** Blends a picture with another picture.

Notes:

If DestImage is nil, BackgroundColour is used for the background.
If no mask is specified, a full black mask is used.

Result must be a 24bit or 32bit picture.
See also:

- 2.1 BlendPicturesWithMaskWithBackgroundMBS(SourceImage As Picture, DestImage As Picture, Mask As Picture, Result As Picture, X As Integer, Y As Integer, Width As Integer, Height As Integer) as boolean 18

ColorizePictureMBS(Pict As Picture, Mask As Picture, foreR as double, foreG as double, foreB as double, foreA as double, backR as double, backG as double, backB as double, backA as double) as boolean

Plugin Version: 10.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: No, . **Function:** Colorizes a picture.

Example:

```
dim p as Picture = SpecialFolder.Pictures.Child("test2.tif").OpenAsPicture

if ColorizePictureMBS(p, p.mask, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.1) then
  Backdrop=p
end if
```

Notes:

The given pictures are edited. As editing pictures works only on Mac and Windows if the pictures are 24 or 32 bit, this does not work on Linux.

Returns true on success and false on failure.

CombinePicturesMBS(red as picture, blue as picture, green as picture) as picture

Plugin Version: 4.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Combines the red, green and blue channels of three images into the a new one.

Notes: Returns nil on any error.

DiffPicturesMBS(source as picture, dest as picture, square as boolean) as picture

Plugin Version: 11.1 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Calculates the difference between two pictures.

Example:

```
// our test Picture
dim p as Picture = LogoMBS(500)

// compress with JPEG and 10%
dim d as string = PictureToJPEGStringMBS(p, 10)

// decompress
```

```
dim q as Picture = JPEGStringToPictureMBS(d, true)
```

```
// compare them
window1.Backdrop = DiffPicturesMBS(p, q, true)
```

Notes:

Source and dest pictures must have same size. If square, the error is squared, so you see it much better.

Returns nil in case not enough memory is available or pictures do not have same size or are nil.

If both pictures are equal, all pixels in the returned picture are black.
See also `Picture.isBlackMBS`, and `Picture.CompareMBS`.

GetMBfromPictureMBS(pic as picture, mask as picture, mode as string) as memoryblock

Plugin Version: 10.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a memoryblock from the picture data with the given format.

Example:

```
dim p as Picture = LogoMBS(500)
dim m as MemoryBlock = GetMBfromPictureMBS(p, p.mask, "RGB32")
```

Notes:

Returns nil on any error.

Mode can be a string with the following strings: RGB16, ARGB16, RGB16_565, ARGB32, RGB32, RGB24 or MASK8.

See the example project "Picture To Memoryblock.rbp" for the RB code matching the plugin code.

See also:

- 2.1 GetMBfromPictureMBS(pic as picture, mode as string) as memoryblock

GetMBfromPictureMBS(pic as picture, mode as string) as memoryblock

Plugin Version: 10.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a memoryblock from the picture data with the given format.

Example:

```
dim p as Picture = LogoMBS(500)
dim m as MemoryBlock = GetMBfromPictureMBS(p, "RGB32")
```

Notes: Same as the other GetMBfromPictureMBS function, but takes the mask from the picture.
See also:

- 2.1 GetMBfromPictureMBS(pic as picture, mask as picture, mode as string) as memoryblock 20

MandelbrotSetMBS(Threaded as integer, width as integer, height as integer, fx as double = 4.0, fy as double = 4.0, dx as double = -2.0, dy as double = -2.0) as picture

Plugin Version: 10.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Calculates the mandelbrot picture.

Example:

```
Backdrop = MandelbrotSetMBS(0,300,300)
```

Notes:

Threaded parameter specifies how many threads you want to use:

A negative value disables threading, zero will use one thread for each CPU core and a positive number specifies the thread count.

Width & Height specify the output image size.

fx and fy are the scale values and dx/dy specify the the position of the mandelbrot image.

MemoryblockABGRtoPictureMBS(dest as picture, source as memoryblock, offset as integer, width as integer, height as integer) as picture

Plugin Version: 10.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a memoryblock into a picture object.

Notes:

Returns nil on any error.
source should not be nil.
offset should be 0 or bigger and is the start position in the memoryblock.

The function will crash if the memoryblock is too small. Needs width*height*4 bytes in the memoryblock.

In the dest picture parameter you can provide a picture to draw in. If the picture is no big enough or nil, a new one is created.

Does not access the mask inside the image!
See also:

- 2.1 MemoryblockABGRtoPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer) as picture 22

MemoryblockABGRtoPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer) as picture

Plugin Version: 8.5 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a memoryblock into a picture object.

Notes:

Returns nil on any error.
source should not be nil.
offset should be 0 or bigger and is the start position in the memoryblock.

The function will crash if the memoryblock is too small. Needs width*height*4 bytes in the memoryblock.

Does not access the mask inside the image!
See also:

- 2.1 MemoryblockABGRtoPictureMBS(dest as picture, source as memoryblock, offset as integer, width as integer, height as integer) as picture 22

MemoryblockARGBtoPictureMBS(dest as picture, source as memoryblock, offset as integer, width as integer, height as integer) as picture

Plugin Version: 10.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a memoryblock into a picture object.

Example:

```

dim m as MemoryBlock
dim p,q as Picture

p=NewPicture(100,100,32)
p.Graphics.ForeColor=rgb(255,128,1)
p.Graphics.FillRect 0,0,100,100

// Make a new MemoryBlock
m=NewMemoryBlock(100*100*4) // 3 bytes per Pixel

// Copy RGB without alpha
if p.CopyARGBtoMemoryblockMBS(m,0,0) then

dim x as Picture = NewPicture(100,100,32)

q=MemoryblockARGBtoPictureMBS(x, m,0,100,100)

Backdrop=q

if x=q then
window1.Title = "reused picture"
else
window1.Title = "created new picture"
end if
end if

```

Notes:

Returns nil on any error.

source should not be nil.

offset should be 0 or bigger and is the start position in the memoryblock.

The function will crash if the memoryblock is too small. Needs width*height*4 bytes in the memoryblock.

In the dest picture parameter you can provide a picture to draw in. If the picture is no big enough or nil, a new one is created.

Does not access the mask inside the image!

See also:

- 2.1 MemoryblockARGBtoPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer) as picture 24

MemoryblockARGBtoPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer) as picture

Plugin Version: 8.5 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a memoryblock into a picture object.

Notes:

Returns nil on any error.

source should not be nil.

offset should be 0 or bigger and is the start position in the memoryblock.

The function will crash if the memoryblock is too small. Needs width*height*4 bytes in the memoryblock.

Does not access the mask inside the image!

See also:

- 2.1 MemoryblockARGBtoPictureMBS(dest as picture, source as memoryblock, offset as integer, width as integer, height as integer) as picture 23

MemoryblockARGBtoPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer, LittleEndian as boolean) as picture

Plugin Version: 6.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a memoryblock into a picture object.

Example:

```

const kAlphaOffset=0 ' (BigEndian) and 3 (LittleEndian)
dim m as MemoryBlock
dim p,q,k as Picture

p=NewPicture(100,100,32)
p.Graphics.ForeColor=rgb(255,128,1)
p.Graphics.FillRect 0,0,100,100
p.mask.Graphics.ForeColor=rgb(127,127,127)
p.mask.Graphics.FillRect 0,0,100,100

// Make a new MemoryBlock
m=NewMemoryBlock(100*100*4) // 4 bytes per Pixel

// copy RGB and leave room for alpha
if p.CopyARGBtoMemoryblockMBS(m,0,false,-1) then
'MsgBox EncodingToHexMBS(m.StringValue(0,99))
end if

// copy green channel from mask image into Memoryblock
if p.mask.CopyGtoMemoryblockMBS(m,kAlphaOffset,4) then
'MsgBox EncodingToHexMBS(m.StringValue(0,99))
end if

// make the picture from this Memoryblock
q=MemoryblockARGBtoPictureMBS(m,0,100,100,false)

// make the mask from this Memoryblock
k=MemoryblockGrayToPictureMBS(m,kAlphaOffset,100,100,4)

// combine picture and mask
q.Mask.Graphics.DrawPicture k,0,0

Backdrop=q

```

Notes:

Returns nil on any error.

source should not be nil.

offset should be 0 or bigger and is the start position in the memoryblock.

The function will crash if the memoryblock is too small. Needs width*height*4 bytes in the memoryblock.

Does not access the mask inside the image!

LittleEndian specifies whether the image is stored in ARGB (BigEndian) or BGRA (LittleEndian) mode.

MemoryblockBGRAtoPictureMBS(dest as picture, source as memoryblock, offset as integer, width as integer, height as integer) as picture

Plugin Version: 10.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a memoryblock into a picture object.

Notes:

Returns nil on any error.

source should not be nil.

offset should be 0 or bigger and is the start position in the memoryblock.

The function will crash if the memoryblock is too small. Needs width*height*4 bytes in the memoryblock.

In the dest picture parameter you can provide a picture to draw in. If the picture is no big enough or nil, a new one is created.

Does not access the mask inside the image!

See also:

- 2.1 MemoryblockBGRAtoPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer) as picture 26

MemoryblockBGRAtoPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer) as picture

Plugin Version: 8.5 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a memoryblock into a picture object.

Notes:

Returns nil on any error.

source should not be nil.

offset should be 0 or bigger and is the start position in the memoryblock.

The function will crash if the memoryblock is too small. Needs width*height*4 bytes in the memoryblock.

Does not access the mask inside the image!
See also:

- 2.1 MemoryblockBGRtoPictureMBS(dest as picture, source as memoryblock, offset as integer, width as integer, height as integer) as picture 26

MemoryblockBGRtoPictureMBS(dest as picture, source as memoryblock, offset as integer, width as integer, height as integer) as picture

Plugin Version: 10.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a memoryblock into a picture object.

Example:

```

dim m as MemoryBlock
dim p,q as Picture

p=NewPicture(100,100,32)
p.Graphics.ForeColor=rgb(255,128,1)
p.Graphics.FillRect 0,0,100,100

// Make a new MemoryBlock
m=NewMemoryBlock(100*100*3) // 3 bytes per Pixel

// Copy RGB without alpha
if p.CopyBGRtoMemoryblockMBS(m,0) then

dim x as Picture = NewPicture(100,100,32)

q=MemoryblockBGRtoPictureMBS(x, m,0,100,100)

Backdrop=q

if x=q then
window1.Title = "reused picture"
else
window1.Title = "created new picture"
end if

end if

```

Notes:

Returns nil on any error.
 source should not be nil.
 offset should be 0 or bigger and is the start position in the memoryblock.

The function will crash if the memoryblock is too small. Needs width*height*3 bytes in the memoryblock.

In the dest picture parameter you can provide a picture to draw in. If the picture is no big enough or nil, a new one is created.

Does not access the mask inside the image!
 See also:

- 2.1 MemoryblockBGRtoPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer) as picture 28

MemoryblockBGRtoPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer) as picture

Plugin Version: 8.5 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a memoryblock into a picture object.

Notes:

Returns nil on any error.
 source should not be nil.
 offset should be 0 or bigger and is the start position in the memoryblock.

The function will crash if the memoryblock is too small. Needs width*height*3 bytes in the memoryblock.

Does not access the mask inside the image!
 See also:

- 2.1 MemoryblockBGRtoPictureMBS(dest as picture, source as memoryblock, offset as integer, width as integer, height as integer) as picture 27

MemoryblockGrayToPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer, PixelByteSize as integer) as picture

Plugin Version: 6.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a memoryblock into a picture object.

Example:

```

const kAlphaOffset=0 ' (BigEndian) and 3 (LittleEndian)
dim m as MemoryBlock
dim p,q,k as Picture

p=NewPicture(100,100,32)
p.Graphics.ForeColor=rgb(255,128,1)
p.Graphics.FillRect 0,0,100,100
p.mask.Graphics.ForeColor=rgb(127,127,127)
p.mask.Graphics.FillRect 0,0,100,100

// Make a new MemoryBlock
m=NewMemoryBlock(100*100*4) // 4 bytes per Pixel

// copy RGB and leave room for alpha
if p.CopyARGBtoMemoryblockMBS(m,0,false,-1) then
'MsgBox EncodingToHexMBS(m.StringValue(0,99))
end if

// copy green channel from mask image into Memoryblock
if p.mask.CopyGtoMemoryblockMBS(m,kAlphaOffset,4) then
'MsgBox EncodingToHexMBS(m.StringValue(0,99))
end if

// make the picture from this Memoryblock
q=MemoryblockARGBtoPictureMBS(m,0,100,100,false)

// make the mask from this Memoryblock
k=MemoryblockGrayToPictureMBS(m,kAlphaOffset,100,100,4)

// combine picture and mask
q.Mask.Graphics.DrawPicture k,0,0

Backdrop=q

```

Notes:

Returns nil on any error.

source should not be nil.

offset should be 0 or bigger and is the start position in the memoryblock.

The function will crash if the memoryblock is too small. Needs width*height*PixelByteSize bytes in the memoryblock.

Does not access the mask inside the image!

See also:

- 2.1 MemoryblockGrayToPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer, PixelByteSize as integer, Red as integer, Blue as integer, Green as integer) as picture 30
- 2.1 MemoryblockGrayToPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer, PixelByteSize as integer, Red() as integer, Blue() as integer, Green() as integer) as picture 31

MemoryblockGrayToPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer, PixelByteSize as integer, Red as integer, Blue as integer, Green as integer) as picture

Plugin Version: 8.5 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a memoryblock into a picture object.

Notes:

This variation of this method Multiplies the gray value with Red, Blue and Green and divided by 256.

Returns nil on any error.

source should not be nil.

offset should be 0 or bigger and is the start position in the memoryblock.

The function will crash if the memoryblock is too small. Needs width*height*PixelByteSize bytes in the memoryblock.

Does not access the mask inside the image!

See also:

- 2.1 MemoryblockGrayToPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer, PixelByteSize as integer) as picture 29

- 2.1 MemoryblockGrayToPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer, PixelByteSize as integer, Red() as integer, Blue() as integer, Green() as integer) as picture 31

MemoryblockGrayToPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer, PixelByteSize as integer, Red() as integer, Blue() as integer, Green() as integer) as picture

Plugin Version: 8.5 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a memoryblock into a picture object.

Notes:

This variation of this method lookups the Red, Green and Blue values for the next pixel by using the gray value as index.

The arrays should have 256 elements.

Returns nil on any error.

source should not be nil.

offset should be 0 or bigger and is the start position in the memoryblock.

The function will crash if the memoryblock is too small. Needs width*height*PixelByteSize bytes in the memoryblock.

Does not access the mask inside the image!

See also:

- 2.1 MemoryblockGrayToPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer, PixelByteSize as integer) as picture 29
- 2.1 MemoryblockGrayToPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer, PixelByteSize as integer, Red as integer, Blue as integer, Green as integer) as picture 30

MemoryblockRGBAToPictureMBS(dest as picture, source as memoryblock, offset as integer, width as integer, height as integer, FlipVertically as boolean=false) as picture

Plugin Version: 10.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a memoryblock into a picture object.

Notes:

Returns nil on any error.
 source should not be nil.
 offset should be 0 or bigger and is the start position in the memoryblock.

The function will crash if the memoryblock is too small. Needs width*height*4 bytes in the memoryblock.

In the dest picture parameter you can provide a picture to draw in. If the picture is no big enough or nil, a new one is created.

If FlipVertically is true the image is flipped. New in version 9.4.

Does not access the mask inside the image!
 See also:

- 2.1 MemoryBlockRGBAToPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer, FlipVertically as boolean=false) as picture 32

MemoryBlockRGBAToPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer, FlipVertically as boolean=false) as picture

Plugin Version: 8.5 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a memoryblock into a picture object.

Example:

```
// some memory with pixel data
dim m as MemoryBlock = NewMemoryBlock(100*100*32)

for i as integer = 1 to 1000
// place random pixels
m.Int8Value(rnd*m.size) = rnd*256
next

// and make a picture
dim l as Picture = MemoryBlockRGBAToPictureMBS(m, 0, 100, 100)

// display in window
window1.backdrop = l
```

Notes:

Returns nil on any error.

source should not be nil.

offset should be 0 or bigger and is the start position in the memoryblock.

The function will crash if the memoryblock is too small. Needs width*height*4 bytes in the memoryblock.

If FlipVertically is true the image is flipped. New in version 9.4.

Does not access the mask inside the image!

See also:

- 2.1 MemoryblockRGBtoPictureMBS(dest as picture, source as memoryblock, offset as integer, width as integer, height as integer, FlipVertically as boolean=false) as picture 31

MemoryblockRGBtoPictureMBS(dest as picture, source as memoryblock, offset as integer, width as integer, height as integer) as picture

Plugin Version: 10.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a memoryblock into a picture object.

Example:

```

dim m as MemoryBlock
dim p,q as Picture

p=NewPicture(100,100,32)
p.Graphics.ForeColor=rgb(255,128,1)
p.Graphics.FillRect 0,0,100,100

// Make a new MemoryBlock
m=NewMemoryBlock(100*100*3) // 3 bytes per Pixel

// Copy RGB without alpha
if p.CopyRGBtoMemoryblockMBS(m,0) then

dim x as Picture = NewPicture(100,100,32)

q=MemoryblockRGBtoPictureMBS(x, m,0,100,100)

Backdrop=q

```

```

if x=q then
window1.Title = "reused picture"
else
window1.Title = "created new picture"
end if
end if

```

Notes:

Returns nil on any error.

source should not be nil.

offset should be 0 or bigger and is the start position in the memoryblock.

In the dest picture parameter you can provide a picture to draw in. If the picture is no big enough or nil, a new one is created.

The function will crash if the memoryblock is too small. Needs width*height*3 bytes in the memoryblock.

Does not access the mask inside the image!

MemoryblockRGBtoPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer) as picture

Plugin Version: 6.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a memoryblock into a picture object.

Example:

```

dim m as MemoryBlock
dim p,q as Picture

p=NewPicture(100,100,32)
p.Graphics.ForeColor=rgb(255,128,1)
p.Graphics.FillRect 0,0,100,100

// Make a new MemoryBlock
m=NewMemoryBlock(100*100*3) // 3 bytes per Pixel

// Copy RGB without alpha

```

```

if p.CopyRGBtoMemoryblockMBS(m,0) then
q=MemoryblockRGBtoPictureMBS(m,0,100,100)

Backdrop=q

end if

```

Notes:

Returns nil on any error.
source should not be nil.
offset should be 0 or bigger and is the start position in the memoryblock.

The function will crash if the memoryblock is too small. Needs width*height*3 bytes in the memoryblock.

Does not access the mask inside the image!

NewBluePaletteMBS as PaletteMBS

Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns a 256 color PaletteMBS with blue shades.

Example:

```

dim pal as PaletteMBS = NewBluePaletteMBS
dim p as Picture = NewPicture(512,256,32)
dim g as Graphics = p.Graphics

for x as integer=0 to 255
g.ForeColor=pal.Col(x)
g.FillRect x*2, 0, 2, g.Height
next

Backdrop=p

```

NewGrayPaletteMBS as PaletteMBS

Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns a 256 color PaletteMBS with grays.

Example:

```
dim pal as PaletteMBS = NewGrayPaletteMBS
dim p as Picture = NewPicture(512,256,32)
dim g as Graphics = p.Graphics
```

```
for x as integer=0 to 255
g.ForeColor=pal.Col(x)
g.FillRect x*2, 0, 2, g.Height
next
```

Backdrop=p

NewGreenPaletteMBS as PaletteMBS

Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns a 256 color PaletteMBS with green shades.

Example:

```
dim pal as PaletteMBS = NewGreenPaletteMBS
dim p as Picture = NewPicture(512,256,32)
dim g as Graphics = p.Graphics
```

```
for x as integer=0 to 255
g.ForeColor=pal.Col(x)
g.FillRect x*2, 0, 2, g.Height
next
```

Backdrop=p

NewPaletteMBS(count as integer) as PaletteMBS

Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Makes a new PaletteMBS with the given number of colors.

Notes:

```
Same as:  
p=new PaletteMBS  
p.count=count  
return p
```

NewPalmPaletteMBS as PaletteMBS

Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the Palm system color PaletteMBS.

Example:

```
dim pal as PaletteMBS = NewPalmPaletteMBS  
dim p as Picture = NewPicture(512,256,32)  
dim g as Graphics = p.Graphics  
  
for x as integer=0 to 255  
g.ForeColor=pal.Col(x)  
g.FillRect x*2, 0, 2, g.Height  
next
```

Backdrop=p

Notes: If you want to make 256 color pictures for use on a Palm, you may need that on a Mac.

NewPictureEditorMBS(pic as picture) as PictureEditorMBS

Plugin Version: 11.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new picture editor editing the given picture.

Example:

```
dim l as Picture = LogoMBS(500)  
dim p as PictureEditorMBS  
  
p = NewPictureEditorMBS(l)
```

Notes:

Returns nil on failure.
Works only for bitmap images.

NewPictureMBS(width as integer, height as integer, pixeltype as integer, buffer as memory-block, rowbytes as integer) as picture

Plugin Version: 8.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a picture from a memory block.

Notes:

rowbytes must be the number of bytes per row. Typical width*3 or width*4.

Pixeltype constants:

kRBPixelRGB24	= 1	3 bytes/pixel: Red, Green, Blue
kRBPixelBGR24	= 2	3 bytes/pixel: Blue, Green, Red
kRBPixelXRGB32	= 3	4 bytes/pixel: Unused, Red, Green, Blue
kRBPixelBGRX32	= 4	4 bytes/pixel: Blue, Green, Red, Unused

NewPictureReaderMBS(pic as picture) as PictureReaderMBS

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new picture reader.

Example:

```
dim pic as Picture = LogoMBS(500)
dim p as PictureReaderMBS
dim m as MemoryBlock
dim r,g,b,rRow,gRow,bRow,h1,w1,x,y,bpp as integer

// Create a new picture reader
p=NewPictureReaderMBS(pic)

h1=p.Height-1
w1=p.Width-1

bpp=p.BytesPerPixel
rRow=p.RedOffset
gRow=p.GreenOffset
bRow=p.BlueOffset
// in each row the red, blue and green channels have different offsets.
```

```
// but offsets are platform dependent

dim sum as Double

for y=0 to h1
  // Get data in memory. This Memoryblock has a size property of 0!
  m=p.Data(y)
  r=rRow
  g=gRow
  b=bRow

  for x=0 to w1

    sum = sum + m.UInt8Value(r)
    sum = sum + m.UInt8Value(g)
    sum = sum + m.UInt8Value(b)

    r=r+bpp
    g=g+bpp
    b=b+bpp
  next

next

// show the sum of all pixels:
MsgBox "Sum with plugin is: "+str(sum)

// now try same in RB code:

dim surface as RGBSurface = pic.RGBSurface
dim c as color

sum = 0.0

for y=0 to h1
  for x=0 to w1
    c = surface.Pixel(x,y)

    sum = sum + c.red
    sum = sum + c.Green
    sum = sum + c.Blue

  next

next

surface = nil
```

```
MsgBox "Sum with RB Code is: "+str(sum)
quit
```

Notes:

Returns nil on failure.

Please report if nil is returned as it should work always (except for low memory).

NewPictureWithColorMBS(width as integer, height as integer, c as color) as picture

Plugin Version: 11.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new picture and fills it with the given color.

Example:

```
window1.backdrop = NewPictureWithColorMBS(200, 200, & c3366CC)
```

Notes: This function is mostly to check if the picture writer code in our plugins work.

NewPictureWriterMBS(width as integer, height as integer) as PictureWriterMBS

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new picture writer.

Example:

```
dim p as PictureWriterMBS
dim m as MemoryBlock
dim r,g,b,rRow,gRow,bRow,h1,w1,x,y,bpp as integer
```

```
// Create a new picture writer
p=NewPictureWriterMBS(512,512)
```

```
h1=p.Height-1
w1=p.Width-1
```

```
bpp=p.BytesPerPixel
rRow=p.RedOffset
```

```

gRow=p.GreenOffset
bRow=p.BlueOffset
// in each row the red, blue and green channels have different offsets.
// but offsets are platform dependent

for y=0 to h1
// Get data in memory. This Memoryblock has a size property of 0!
m=p.Data(y)
r=rRow
g=gRow
b=bRow

for x=0 to w1

m.UInt8Value(r)=x\2
m.UInt8Value(g)=y\2
m.UInt8Value(b)=x*y\2

r=r+bpp
g=g+bpp
b=b+bpp
next

next

// Use Render to make a picture object
dim pic as Picture = p.Render
Backdrop = pic

```

Notes: Returns nil on failure (low memory).

NewRedPaletteMBS as PaletteMBS

Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns a 256 color PaletteMBS with red shades.

Example:

```

dim pal as PaletteMBS = NewRedPaletteMBS
dim p as Picture = NewPicture(512,256,32)
dim g as Graphics = p.Graphics

for x as integer=0 to 255

```

```
g.ForeColor=pal.Col(x)
g.FillRect x*2, 0, 2, g.Height
next
```

```
Backdrop=p
```

NewSystemPaletteMBS as PaletteMBS

Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the system color PaletteMBS.

Example:

```
dim pal as PaletteMBS = NewSystemPaletteMBS
dim p as Picture = NewPicture(512,256,32)
dim g as Graphics = p.Graphics
```

```
for x as integer=0 to 255
g.ForeColor=pal.Col(x)
g.FillRect x*2, 0, 2, g.Height
next
```

```
Backdrop=p
```

Notes: On Windows it is a hard coded PaletteMBS as you can only read the current PaletteMBS when the user uses 8bit video.

NewWebPaletteMBS as PaletteMBS

Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns a 256 color PaletteMBS with the 216 websave colors.

Example:

```
dim pal as PaletteMBS = NewWebPaletteMBS
dim p as Picture = NewPicture(512,256,32)
dim g as Graphics = p.Graphics
```

```
for x as integer=0 to 255
g.ForeColor=pal.Col(x)
g.FillRect x*2, 0, 2, g.Height
```

```
next
```

```
Backdrop=p
```

Notes: The PaletteMBS has 256 colors, but only 216 are used.

NewWindowsPaletteMBS as PaletteMBS

Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the Windows system color PaletteMBS.

Example:

```
dim pal as PaletteMBS = NewWindowsPaletteMBS
dim p as Picture = NewPicture(512,256,32)
dim g as Graphics = p.Graphics
```

```
for x as integer=0 to 255
g.ForeColor=pal.Col(x)
g.FillRect x*2, 0, 2, g.Height
next
```

```
Backdrop=p
```

Notes: If you want to make 256 color pictures for use on Windows, you may need that on a Mac.

PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean) as boolean

Plugin Version: 8.7 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies pixels from one picture into another picture with some options.

Notes:

Returns true on success and false on failure.

This function has 4 behaviors depending on the parameters:

1. If mask is nil and no ForeColour and MaskColour values are passed, the pixels are copied to the destination picture.
2. But if there is a mask, the pixels are copied with applying the mask.
3. If the mask color is not defined, the pixels are filled with the fore color applying the mask.
4. As the last variation the pixels are copied and the forecolor, the mask color or black is used with the image as the mask. If UseColours parameter is false black is used for this.

Parameters:

Image: the source picture, must not be nil.

Mask: the mask picture, can be nil.

DestX: destination position

DestY: destination position

SourceX: source position

SourceY: source position

Width: width of the area to copy

Height: height of the area to copy

UseColours: whether to use the mask colour.

ForeColour: the fore colour, optional, can be integer or color

MaskColour: the mask color, optional, can be integer or color

This function is 5 times in the plugin defined to implement having the last two parameters optional and either integer or color. You can pass a negative number for MaskColour or ForeColour to disable this parameter.

The destination image (self) can be either 24 bit or 32 bit.

See also:

- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color) as boolean 45
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color, MaskColour As color) as boolean 48
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer,

- UseColours As Boolean, ForeColour As Integer) as boolean 49
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer, MaskColour As Integer) as boolean 51
 - 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean) as boolean 54
 - 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color) as boolean 56
 - 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color, MaskColour As color) as boolean 58
 - 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer) as boolean 60
 - 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer, MaskColour As Integer) as boolean 62

PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color) as boolean

Plugin Version: 8.7 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies pixels from one picture into another picture with some options.

Example:

```
dim DestImage As Picture
dim Image As Picture
dim Mask As Picture
dim DestX As Integer=100
dim DestY As Integer=100
dim SourceX As Integer=0
dim SourceY As Integer=0
dim Width As Integer=500
dim Height As Integer=500
```

```
image=LogoMBS(500)
Mask=nil
```

```
DestImage=NewPicture(700,700,32)
```

```
if PictureCombineMBS(DestImage, image, Mask, DestX, DestY, SourceX, SourceY, Width, Height, true,&
c7777777,& c7777777) then
window1.Backdrop=DestImage
end if
```

Notes:

Returns true on success and false on failure.

This function has 4 behaviors depending on the parameters:

1. If mask is nil and no ForeColour and MaskColour values are passed, the pixels are copied to the destination picture.
2. But if there is a mask, the pixels are copied with applying the mask.
3. If the mask color is not defined, the pixels are filled with the fore color applying the mask.
4. As the last variation the pixels are copied and the forecolor, the mask color or black is used with the image as the mask. If UseColours parameter is false black is used for this.

Parameters:

Image: the source picture, must not be nil.

Mask: the mask picture, can be nil.

DestX: destination position

DestY: destination position

SourceX: source position

SourceY: source position

Width: width of the area to copy

Height: height of the area to copy

UseColours: whether to use the mask colour.

ForeColour: the fore colour, optional, can be integer or color

MaskColour: the mask color, optional, can be integer or color

This function is 5 times in the plugin defined to implement having the last two parameters optional and either

integer or color. You can pass a negative number for MaskColour or ForeColour to disable this parameter.

The destination image (self) can be either 24 bit or 32 bit.

See also:

- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean) as boolean 43
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color, MaskColour As color) as boolean 48
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer) as boolean 49
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer, MaskColour As Integer) as boolean 51
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean) as boolean 54
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color) as boolean 56
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color, MaskColour As color) as boolean 58
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer) as boolean 60
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer, MaskColour As Integer) as boolean 62

PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color, MaskColour As color) as boolean

Plugin Version: 8.7 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies pixels from one picture into another picture with some options.

Notes:

Returns true on success and false on failure.

This function has 4 behaviors depending on the parameters:

1. If mask is nil and no ForeColour and MaskColour values are passed, the pixels are copied to the destination picture.
2. But if there is a mask, the pixels are copied with applying the mask.
3. If the mask color is not defined, the pixels are filled with the fore color applying the mask.
4. As the last variation the pixels are copied and the forecolor, the mask color or black is used with the image as the mask. If UseColours parameter is false black is used for this.

Parameters:

Image: the source picture, must not be nil.

PreMultipliedSource: Optional parameter. If true the image must be premultiplied. Default is false.

Mask: the mask picture, can be nil.

DestX: destination position

DestY: destination position

SourceX: source position

SourceY: source position

Width: width of the area to copy

Height: height of the area to copy

UseColours: whether to use the mask colour.

ForeColour: the fore colour, optional, can be integer or color

MaskColour: the mask color, optional, can be integer or color

This function is 5 times in the plugin defined to implement having the last two parameters optional and either integer or color. You can pass a negative number for MaskColour or ForeColour to disable this parameter.

The destination image (self) can be either 24 bit or 32 bit.

See also:

- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean) as boolean 43
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color) as boolean 45
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer) as boolean 49
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer, MaskColour As Integer) as boolean 51
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean) as boolean 54
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color) as boolean 56
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color, MaskColour As color) as boolean 58
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer) as boolean 60
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer, MaskColour As Integer) as boolean 62

PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer) as boolean

Plugin Version: 8.7 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies pixels from one picture into another picture with some options.

Notes:

Returns true on success and false on failure.

This function has 4 behaviors depending on the parameters:

1. If mask is nil and no ForeColour and MaskColour values are passed, the pixels are copied to the destination picture.
2. But if there is a mask, the pixels are copied with applying the mask.
3. If the mask color is not defined, the pixels are filled with the fore color applying the mask.
4. As the last variation the pixels are copied and the forecolor, the mask color or black is used with the image as the mask. If UseColours parameter is false black is used for this.

Parameters:

Image: the source picture, must not be nil.

PreMultipliedSource: Optional parameter. If true the image must be premultiplied. Default is false.

Mask: the mask picture, can be nil.

DestX: destination position

DestY: destination position

SourceX: source position

SourceY: source position

Width: width of the area to copy

Height: height of the area to copy

UseColours: whether to use the mask colour.

ForeColour: the fore colour, optional, can be integer or color

MaskColour: the mask color, optional, can be integer or color

This function is 5 times in the plugin defined to implement having the last two parameters optional and either integer or color. You can pass a negative number for MaskColour or ForeColour to disable this parameter.

The destination image (self) can be either 24 bit or 32 bit.

See also:

- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer,

- DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean) as boolean 43
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color) as boolean 45
 - 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color, MaskColour As color) as boolean 48
 - 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer, MaskColour As Integer) as boolean 51
 - 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean) as boolean 54
 - 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color) as boolean 56
 - 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color, MaskColour As color) as boolean 58
 - 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer) as boolean 60
 - 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer, MaskColour As Integer) as boolean 62

PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer, MaskColour As Integer) as boolean

Plugin Version: 8.7 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies pixels from one picture into another picture with some options.

Example:

```
dim DestImage As Picture
dim Image As Picture
dim Mask As Picture
```

```

dim DestX As Integer=100
dim DestY As Integer=100
dim SourceX As Integer=0
dim SourceY As Integer=0
dim Width As Integer=500
dim Height As Integer=500

image=LogoMBS(500)
Mask=nil
DestImage=NewPicture(700,700,32)

if PictureCombineMBS(DestImage,image,Mask,DestX,DestY,SourceX,SourceY,Width,Height,true,& h777777,&
h777777) then
window1.Backdrop=DestImage
end if

```

Notes:

Returns true on success and false on failure.

This function has 4 behaviors depending on the parameters:

1. If mask is nil and no ForeColour and MaskColour values are passed, the pixels are copied to the destination picture.
2. But if there is a mask, the pixels are copied with applying the mask.
3. If the mask color is not defined, the pixels are filled with the fore color applying the mask.
4. As the last variation the pixels are copied and the forecolor, the mask color or black is used with the image as the mask. If UseColours parameter is false black is used for this.

Parameters:

Image: the source picture, must not be nil.

Mask: the mask picture, can be nil.

DestX: destination position

DestY: destination position

SourceX: source position

SourceY: source position
 Width: width of the area to copy
 Height: height of the area to copy
 UseColours: whether to use the mask colour.
 ForeColour: the fore colour, optional, can be integer or color
 MaskColour: the mask color, optional, can be integer or color

This function is 5 times in the plugin defined to implement having the last two parameters optional and either integer or color. You can pass a negative number for MaskColour or ForeColour to disable this parameter.

The destination image (self) can be either 24 bit or 32 bit.
 See also:

- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean) as boolean 43
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color) as boolean 45
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color, MaskColour As color) as boolean 48
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer) as boolean 49
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean) as boolean 54
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color) as boolean 56
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color, MaskColour As color) as boolean 58
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer) as boolean 60
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width

As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer, MaskColour As Integer)
as boolean 62

PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean) as boolean

Plugin Version: 9.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies pixels from one picture into another picture with some options.

Notes:

Returns true on success and false on failure.

This function has 4 behaviors depending on the parameters:

1. If mask is nil and no ForeColour and MaskColour values are passed, the pixels are copied to the destination picture.
2. But if there is a mask, the pixels are copied with applying the mask.
3. If the mask color is not defined, the pixels are filled with the fore color applying the mask.
4. As the last variation the pixels are copied and the forecolor, the mask color or black is used with the image as the mask. If UseColours parameter is false black is used for this.

Parameters:

Image: the source picture, must not be nil.

PreMultipliedSource: Optional parameter. If true the image must be premultiplied. Default is false.

Mask: the mask picture, can be nil.

DestX: destination position

DestY: destination position

SourceX: source position

SourceY: source position

Width: width of the area to copy

Height: height of the area to copy

UseColours: whether to use the mask colour.

ForeColour: the fore colour, optional, can be integer or color

MaskColour: the mask color, optional, can be integer or color

This function is 5 times in the plugin defined to implement having the last two parameters optional and either integer or color. You can pass a negative number for MaskColour or ForeColour to disable this parameter.

The destination image (self) can be either 24 bit or 32 bit.
See also:

- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean) as boolean 43
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color) as boolean 45
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color, MaskColour As color) as boolean 48
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer) as boolean 49
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer, MaskColour As Integer) as boolean 51
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color) as boolean 56
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color, MaskColour As color) as boolean 58
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer) as boolean 60
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer, MaskColour As Integer) as boolean 62

PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color) as boolean

Plugin Version: 9.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies pixels from one picture into another picture with some options.

Notes:

Returns true on success and false on failure.

This function has 4 behaviors depending on the parameters:

1. If mask is nil and no ForeColour and MaskColour values are passed, the pixels are copied to the destination picture.
2. But if there is a mask, the pixels are copied with applying the mask.
3. If the mask color is not defined, the pixels are filled with the fore color applying the mask.
4. As the last variation the pixels are copied and the forecolor, the mask color or black is used with the image as the mask. If UseColours parameter is false black is used for this.

Parameters:

Image: the source picture, must not be nil.

PreMultipliedSource: Optional parameter. If true the image must be premultiplied. Default is false.

Mask: the mask picture, can be nil.

DestX: destination position

DestY: destination position

SourceX: source position

SourceY: source position

Width: width of the area to copy

Height: height of the area to copy

UseColours: whether to use the mask colour.

ForeColour: the fore colour, optional, can be integer or color

MaskColour: the mask color, optional, can be integer or color

This function is 5 times in the plugin defined to implement having the last two parameters optional and either

integer or color. You can pass a negative number for MaskColour or ForeColour to disable this parameter.

The destination image (self) can be either 24 bit or 32 bit.

See also:

- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean) as boolean 43
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color) as boolean 45
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color, MaskColour As color) as boolean 48
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer) as boolean 49
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer, MaskColour As Integer) as boolean 51
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean) as boolean 54
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color, MaskColour As color) as boolean 58
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer) as boolean 60
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer, MaskColour As Integer) as boolean 62

PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color, MaskColour As color) as boolean

Plugin Version: 9.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies pixels from one picture into another picture with some options.

Notes:

Returns true on success and false on failure.

This function has 4 behaviors depending on the parameters:

1. If mask is nil and no ForeColour and MaskColour values are passed, the pixels are copied to the destination picture.
2. But if there is a mask, the pixels are copied with applying the mask.
3. If the mask color is not defined, the pixels are filled with the fore color applying the mask.
4. As the last variation the pixels are copied and the forecolor, the mask color or black is used with the image as the mask. If UseColours parameter is false black is used for this.

Parameters:

Image: the source picture, must not be nil.

PreMultipliedSource: Optional parameter. If true the image must be premultiplied. Default is false.

Mask: the mask picture, can be nil.

DestX: destination position

DestY: destination position

SourceX: source position

SourceY: source position

Width: width of the area to copy

Height: height of the area to copy

UseColours: whether to use the mask colour.

ForeColour: the fore colour, optional, can be integer or color

MaskColour: the mask color, optional, can be integer or color

This function is 5 times in the plugin defined to implement having the last two parameters optional and either

integer or color. You can pass a negative number for MaskColour or ForeColour to disable this parameter.

The destination image (self) can be either 24 bit or 32 bit.

See also:

- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean) as boolean 43
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color) as boolean 45
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color, MaskColour As color) as boolean 48
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer) as boolean 49
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer, MaskColour As Integer) as boolean 51
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean) as boolean 54
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color) as boolean 56
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer) as boolean 60
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer, MaskColour As Integer) as boolean 62

PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer) as boolean

Plugin Version: 9.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies pixels from one picture into another picture with some options.

Notes:

Returns true on success and false on failure.

This function has 4 behaviors depending on the parameters:

1. If mask is nil and no ForeColour and MaskColour values are passed, the pixels are copied to the destination picture.
2. But if there is a mask, the pixels are copied with applying the mask.
3. If the mask color is not defined, the pixels are filled with the fore color applying the mask.
4. As the last variation the pixels are copied and the forecolor, the mask color or black is used with the image as the mask. If UseColours parameter is false black is used for this.

Parameters:

Image: the source picture, must not be nil.

PreMultipliedSource: Optional parameter. If true the image must be premultiplied. Default is false.

Mask: the mask picture, can be nil.

DestX: destination position

DestY: destination position

SourceX: source position

SourceY: source position

Width: width of the area to copy

Height: height of the area to copy

UseColours: whether to use the mask colour.

ForeColour: the fore colour, optional, can be integer or color

MaskColour: the mask color, optional, can be integer or color

This function is 5 times in the plugin defined to implement having the last two parameters optional and either

integer or color. You can pass a negative number for MaskColour or ForeColour to disable this parameter.

The destination image (self) can be either 24 bit or 32 bit.

See also:

- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean) as boolean 43
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color) as boolean 45
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color, MaskColour As color) as boolean 48
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer) as boolean 49
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer, MaskColour As Integer) as boolean 51
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean) as boolean 54
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color) as boolean 56
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color, MaskColour As color) as boolean 58
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer, MaskColour As Integer) as boolean 62

PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer, MaskColour As Integer) as boolean

Plugin Version: 9.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies pixels from one picture into another picture with some options.

Notes:

Returns true on success and false on failure.

This function has 4 behaviors depending on the parameters:

1. If mask is nil and no ForeColour and MaskColour values are passed, the pixels are copied to the destination picture.
2. But if there is a mask, the pixels are copied with applying the mask.
3. If the mask color is not defined, the pixels are filled with the fore color applying the mask.
4. As the last variation the pixels are copied and the forecolor, the mask color or black is used with the image as the mask. If UseColours parameter is false black is used for this.

Parameters:

Image: the source picture, must not be nil.

PreMultipliedSource: Optional parameter. If true the image must be premultiplied. Default is false.

Mask: the mask picture, can be nil.

DestX: destination position

DestY: destination position

SourceX: source position

SourceY: source position

Width: width of the area to copy

Height: height of the area to copy

UseColours: whether to use the mask colour.

ForeColour: the fore colour, optional, can be integer or color

MaskColour: the mask color, optional, can be integer or color

This function is 5 times in the plugin defined to implement having the last two parameters optional and either

integer or color. You can pass a negative number for MaskColour or ForeColour to disable this parameter.

The destination image (self) can be either 24 bit or 32 bit.

See also:

- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean) as boolean 43
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color) as boolean 45
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color, MaskColour As color) as boolean 48
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer) as boolean 49
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer, MaskColour As Integer) as boolean 51
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean) as boolean 54
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color) as boolean 56
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color, MaskColour As color) as boolean 58
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer) as boolean 60

PictureCopyPixelFastMBS(DestImage As Picture, Source As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer) as boolean

Plugin Version: 8.7 Console & Web: No Mac: Yes, Win: Yes, Linux: No, . **Function:** Copies pixels from one picture into another picture with some options.

Example:

```
const x=100 // mouse coordinates for example
const y=100

dim p,logo as picture

logo=LogoMBS(500)

p=NewPicture(800,800,32)

p.Graphics.ForeColor=&cFFFFFF
p.Graphics.FillRect 0,0,p.Width,p.Height

if PictureCopyPixelFastMBS(p, logo, x-logo.Width/2, y-logo.Height/2, 0, 0, logo.Width, logo.Height) then
' ok
else
beep
end if

window1.Backdrop=p
```

Notes:

Returns true on success and false on failure.

Parameters:

Source: the source picture, must not be nil.

DestX: destination position

DestY: destination position

SourceX: source position

SourceY: source position

Width: width of the area to copy

Height: height of the area to copy

The destination image (self) can be either 24 bit or 32 bit.

The source image can have any bit depth and may be converted to 24 or 32 bit.

PtrABGRtoPictureMBS(dest as picture, source as Ptr, offset as integer, width as integer, height as integer) as picture

Plugin Version: 12.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a pointer into a picture object.

Notes:

Returns nil on any error.

source should not be nil.

offset should be 0 or bigger and is the start position in the data the pointer points to.

The function will crash if the data is too small where the pointer points to. Needs width*height*4 bytes in the memory pointed to by pointer plus offset.

In the dest picture parameter you can provide a picture to draw in. If the picture is no big enough or nil, a new one is created.

Does not access the mask inside the image!

See also:

- 2.1 PtrABGRtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer) as picture 65

PtrABGRtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer) as picture

Plugin Version: 12.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a pointer into a picture object.

Notes:

Returns nil on any error.

source should not be nil.

offset should be 0 or bigger and is the start position in the data the pointer points to.

The function will crash if the data is too small where the pointer points to. Needs width*height*4 bytes in the memory pointed to by pointer plus offset.

Does not access the mask inside the image!

See also:

- 2.1 PtrABGRtoPictureMBS(dest as picture, source as Ptr, offset as integer, width as integer, height as integer) as picture 65

PtrARGBtoPictureMBS(dest as picture, source as Ptr, offset as integer, width as integer, height as integer) as picture

Plugin Version: 12.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a pointer into a picture object.

Notes:

Returns nil on any error.

source should not be nil.

offset should be 0 or bigger and is the start position in the data the pointer points to.

The function will crash if the data is too small where the pointer points to. Needs width*height*4 bytes in the memory pointed to by pointer plus offset.

In the dest picture parameter you can provide a picture to draw in. If the picture is no big enough or nil, a new one is created.

Does not access the mask inside the image!

See also:

- 2.1 PtrARGBtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer) as picture 66
- 2.1 PtrARGBtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer, LittleEndian as boolean) as picture 67

PtrARGBtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer) as picture

Plugin Version: 12.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a pointer into a picture object.

Notes:

Returns nil on any error.

source should not be nil.

offset should be 0 or bigger and is the start position in the data the pointer points to.

The function will crash if the data is too small where the pointer points to. Needs width*height*4 bytes in the memory pointed to by pointer plus offset.

Does not access the mask inside the image!

See also:

- 2.1 PtrARGBtoPictureMBS(dest as picture, source as Ptr, offset as integer, width as integer, height as integer) as picture 66
- 2.1 PtrARGBtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer, LittleEndian as boolean) as picture 67

PtrARGBtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer, LittleEndian as boolean) as picture

Plugin Version: 12.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a pointer into a picture object.

Notes:

Returns nil on any error.

source should not be nil.

offset should be 0 or bigger and is the start position in the data the pointer points to.

The function will crash if the data is too small where the pointer points to. Needs width*height*4 bytes in the memory pointed to by pointer plus offset.

Does not access the mask inside the image!

LittleEndian specifies whether the image is stored in ARGB (BigEndian) or BGRA (LittleEndian) mode.

See also:

- 2.1 PtrARGBtoPictureMBS(dest as picture, source as Ptr, offset as integer, width as integer, height as integer) as picture 66
- 2.1 PtrARGBtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer) as picture 66

PtrBGRAToPictureMBS(dest as picture, source as Ptr, offset as integer, width as integer, height as integer) as picture

Plugin Version: 12.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a pointer into a picture object.

Notes:

Returns nil on any error.

source should not be nil.

offset should be 0 or bigger and is the start position in the data the pointer points to.

The function will crash if the data is too small where the pointer points to. Needs width*height*4 bytes in the memory pointed to by pointer plus offset.

In the dest picture parameter you can provide a picture to draw in. If the picture is no big enough or nil, a new one is created.

Does not access the mask inside the image!

See also:

- 2.1 PtrBGRAToPictureMBS(source as Ptr, offset as integer, width as integer, height as integer) as picture 68

PtrBGRAToPictureMBS(source as Ptr, offset as integer, width as integer, height as integer) as picture

Plugin Version: 12.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a pointer into a picture object.

Notes:

Returns nil on any error.

source should not be nil.

offset should be 0 or bigger and is the start position in the data the pointer points to.

The function will crash if the data is too small where the pointer points to. Needs width*height*4 bytes in the memory pointed to by pointer plus offset.

Does not access the mask inside the image!

See also:

- 2.1 PtrBGRtoPictureMBS(dest as picture, source as Ptr, offset as integer, width as integer, height as integer) as picture 68

PtrBGRtoPictureMBS(dest as picture, source as Ptr, offset as integer, width as integer, height as integer) as picture

Plugin Version: 12.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a pointer into a picture object.

Notes:

Returns nil on any error.

source should not be nil.

offset should be 0 or bigger and is the start position in the data the pointer points to.

The function will crash if the data is too small where the pointer points to. Needs width*height*3 bytes in the memory pointed to by pointer plus offset.

In the dest picture parameter you can provide a picture to draw in. If the picture is no big enough or nil, a new one is created.

Does not access the mask inside the image!

See also:

- 2.1 PtrBGRtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer) as picture 69

PtrBGRtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer) as picture

Plugin Version: 12.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a pointer into a picture object.

Notes:

Returns nil on any error.

source should not be nil.

offset should be 0 or bigger and is the start position in the data the pointer points to.

The function will crash if the data is too small where the pointer points to. Needs width*height*3 bytes in the memory pointed to by pointer plus offset.

Does not access the mask inside the image!
See also:

- 2.1 PtrBGRtoPictureMBS(dest as picture, source as Ptr, offset as integer, width as integer, height as integer) as picture 69

PtrGrayToPictureMBS(source as Ptr, offset as integer, width as integer, height as integer, PixelByteSize as integer) as picture

Plugin Version: 12.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a pointer into a picture object.

Notes:

Returns nil on any error.
source should not be nil.
offset should be 0 or bigger and is the start position in the data the pointer points to.

The function will crash if the data is too small where the pointer points to. Needs width*height*PixelByteSize bytes in the memory pointed to by pointer plus offset.

Does not access the mask inside the image!
See also:

- 2.1 PtrGrayToPictureMBS(source as Ptr, offset as integer, width as integer, height as integer, PixelByteSize as integer, Red as integer, Blue as integer, Green as integer) as picture 70
- 2.1 PtrGrayToPictureMBS(source as Ptr, offset as integer, width as integer, height as integer, PixelByteSize as integer, Red() as integer, Blue() as integer, Green() as integer) as picture 71

PtrGrayToPictureMBS(source as Ptr, offset as integer, width as integer, height as integer, PixelByteSize as integer, Red as integer, Blue as integer, Green as integer) as picture

Plugin Version: 12.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a pointer into a picture object.

Notes:

This variation of this method Multiplies the gray value with Red, Blue and Green and divided by 256.

Returns nil on any error.

source should not be nil.

offset should be 0 or bigger and is the start position in the data the pointer points to.

The function will crash if the data is too small where the pointer points to. Needs width*height*PixelByteSize bytes in the memory pointed to by pointer plus offset.

Does not access the mask inside the image!

See also:

- 2.1 PtrGrayToPictureMBS(source as Ptr, offset as integer, width as integer, height as integer, PixelByteSize as integer) as picture 70
- 2.1 PtrGrayToPictureMBS(source as Ptr, offset as integer, width as integer, height as integer, PixelByteSize as integer, Red() as integer, Blue() as integer, Green() as integer) as picture 71

PtrGrayToPictureMBS(source as Ptr, offset as integer, width as integer, height as integer, PixelByteSize as integer, Red() as integer, Blue() as integer, Green() as integer) as picture

Plugin Version: 12.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a pointer into a picture object.

Notes:

This variation of this method lookups the Red, Green and Blue values for the next pixel by using the gray value as index.

The arrays should have 256 elements.

Returns nil on any error.

source should not be nil.

offset should be 0 or bigger and is the start position in the data the pointer points to.

The function will crash if the data is too small where the pointer points to. Needs width*height*PixelByteSize bytes in the memory pointed to by pointer plus offset.

Does not access the mask inside the image!
See also:

- 2.1 `PtrGrayToPictureMBS`(source as Ptr, offset as integer, width as integer, height as integer, Pixel-ByteSize as integer) as picture 70
- 2.1 `PtrGrayToPictureMBS`(source as Ptr, offset as integer, width as integer, height as integer, Pixel-ByteSize as integer, Red as integer, Blue as integer, Green as integer) as picture 70

`PtrRGBAtoPictureMBS`(dest as picture, source as Ptr, offset as integer, width as integer, height as integer, FlipVertically as boolean=false) as picture

Plugin Version: 12.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a pointer into a picture object.

Notes:

Returns nil on any error.

source should not be nil.

offset should be 0 or bigger and is the start position in the data the pointer points to.

The function will crash if the data is too small where the pointer points to. Needs width*height*4 bytes in the memory pointed to by pointer plus offset.

In the dest picture parameter you can provide a picture to draw in. If the picture is no big enough or nil, a new one is created.

If FlipVertically is true the image is flipped. New in version 9.4.

Does not access the mask inside the image!
See also:

- 2.1 `PtrRGBAtoPictureMBS`(source as Ptr, offset as integer, width as integer, height as integer, FlipVertically as boolean=false) as picture 73

PtrRGBAtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer, FlipVertically as boolean=false) as picture

Plugin Version: 12.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a pointer into a picture object.

Notes:

Returns nil on any error.

source should not be nil.

offset should be 0 or bigger and is the start position in the data the pointer points to.

The function will crash if the data is too small where the pointer points to. Needs width*height*4 bytes in the memory pointed to by pointer plus offset.

If FlipVertically is true the image is flipped. New in version 9.4.

Does not access the mask inside the image!

See also:

- 2.1 PtrRGBAtoPictureMBS(dest as picture, source as Ptr, offset as integer, width as integer, height as integer, FlipVertically as boolean=false) as picture 72

PtrRGBtoPictureMBS(dest as picture, source as Ptr, offset as integer, width as integer, height as integer) as picture

Plugin Version: 12.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a pointer into a picture object.

Notes:

Returns nil on any error.

source should not be nil.

offset should be 0 or bigger and is the start position in the data the pointer points to.

In the dest picture parameter you can provide a picture to draw in. If the picture is no big enough or nil, a new one is created.

The function will crash if the data is too small where the pointer points to. Needs width*height*3 bytes in the memory pointed to by pointer plus offset.

Does not access the mask inside the image!

See also:

- 2.1 PtrRGBtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer) as picture
74

PtrRGBtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer) as picture

Plugin Version: 12.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies image data from a pointer into a picture object.

Notes:

Returns nil on any error.

source should not be nil.

offset should be 0 or bigger and is the start position in the data the pointer points to.

The function will crash if the data is too small where the pointer points to. Needs width*height*3 bytes in the memory pointed to by pointer plus offset.

Does not access the mask inside the image!

See also:

- 2.1 PtrRGBtoPictureMBS(dest as picture, source as Ptr, offset as integer, width as integer, height as integer) as picture
73

TintPictureMBS(source as picture, GreyBase as color, SepiaBase as color) as picture

Plugin Version: 4.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Tints the image.

Example:

// The code does the same thing as this Realbasic code:

```
Sub TintPicture(theImg as Picture, pGreyBase as Color, pSepiaBase as Color)
Dim theRGBSurface as RGBSurface
Dim theWidth, theHeight as Integer
Dim pColor as Color
```

```

Dim x, y as Integer
Dim theGrey as Integer

dim SepiaBaseR as Double
dim SepiaBaseG as Double
dim SepiaBaseB as Double

dim GreyBaseR as Double
dim GreyBaseG as Double
dim GreyBaseB as Double

SepiaBaseR=pSepiaBase.Red / 255.0
SepiaBaseG=pSepiaBase.Green / 255.0
SepiaBaseB=pSepiaBase.Blue / 255.0

GreyBaseR=pGreyBase.Red / 255.0
GreyBaseG=pGreyBase.Green / 255.0
GreyBaseB=pGreyBase.Blue / 255.0

theRGBSurface = theImg.RGBSurface

theWidth = theImg.Width-1
theHeight = theImg.Height-1

For x = 0 to theWidth
For y = 0 to theHeight
pColor = theImg.RGBSurface.Pixel( x, y )

theGrey = ( GreyBaseR * pColor.Red ) + ( GreyBaseG * pColor.Green ) + ( GreyBaseB * pColor.Blue )
theImg.RGBSurface.Pixel( x, y ) = RGB( theGrey * SepiaBaseR, theGrey * SepiaBaseG, theGrey * SepiaBaseB )

Next
Next
End Sub

```

Notes:

You can use the code to do something like a Sepia effect.
Returns a new picture on success.

2.2 class PictureConvolutionMBS

```
class PictureConvolutionMBS
```

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A class for a Picture Convolution.

Example:

```
// blur

dim l as Picture = LogoMBS(500)
dim p as new PictureConvolutionMBS

p.hor(0) = 0.2
p.hor(1) = 0.2
p.hor(2) = 0.2
p.hor(3) = 0.2
p.hor(4) = 0.2

p.ver(0) = 0.2
p.ver(1) = 0.2
p.ver(2) = 0.2
p.ver(3) = 0.2
p.ver(4) = 0.2

p.ValueCount=5

p.SourcePicture=l

dim t as integer=ticks
call p.run(7)
t=ticks-t

Title=str(t)

Backdrop=p.DestinationPicture
```

2.2.1 Methods

close

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The destructor.

Notes:

There is no need to call this method except you want to free all resources of this object now without waiting for Realbasic to do it for you.

(e.g. some Realbasic versions crash on Windows if there are plugin objects not closed.)

Hor(index as integer) as double

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The horizontal factors.

Notes:

Index from 0 to 19.

(Read and Write computed property)

Run(channels as integer) as boolean

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Runs the picture effect.

Notes:

Fails if the pictures are not bitmap pictures. Source and Destination can be equal. If you provide a destination picture, the dimensions of source and destination must be equal.

Channels is a combination of 1, 2 and 4. 1 for Red, 2 for Green and 4 for Blue.

The border (one pixel thick) is not filled in the destination picture.

This method does the following for each pixel:

```
// first horizontal fill the temporary picture
r=0
```

```
g=0
b=0
```

```
if RedChannel then
r = r + sourcepicture.pixel(x-1,y).red * Hor(0)
r = r + sourcepicture.pixel(x ,y).red * Hor(1)
r = r + sourcepicture.pixel(x+1,y).red * Hor(2)
else
r = sourcepicture.pixel(x,y)
end if
```

```
if GreenChannel then
g = g + sourcepicture.pixel(x-1,y).green * Hor(0)
g = g + sourcepicture.pixel(x ,y).green * Hor(1)
g = g + sourcepicture.pixel(x+1,y).green * Hor(2)
else
g = sourcepicture.pixel(x,y)
end if
```

```
if BlueChannel then
b = b + sourcepicture.pixel(x-1,y).blue * Hor(0)
b = b + sourcepicture.pixel(x ,y).blue * Hor(1)
b = b + sourcepicture.pixel(x+1,y).blue * Hor(2)
else
b = sourcepicture.pixel(x,y)
end if
```

```
temppicture.pixel(x,y)=rgb(r,g,b)
```

```
// now back from temporary picture to the destination picture
```

```
r=0
g=0
b=0
```

```
if RedChannel then
r = r + temppicture.pixel(x,y-1).red * Ver(0)
r = r + temppicture.pixel(x,y ).red * Ver(1)
r = r + temppicture.pixel(x,y+1).red * Ver(2)
```

```

else
r = temppicture.pixel(x,y)
end if

if GreenChannel then
g = g + temppicture.pixel(x,y-1).green * Ver(0)
g = g + temppicture.pixel(x,y ).green * Ver(1)
g = g + temppicture.pixel(x,y+1).green * Ver(2)
else
g = temppicture.pixel(x,y)
end if

if BlueChannel then
b = b + temppicture.pixel(x,y-1).blue * Ver(0)
b = b + temppicture.pixel(x,y ).blue * Ver(1)
b = b + temppicture.pixel(x,y+1).blue * Ver(2)
else
b = temppicture.pixel(x,y)
end if

destinationpicture.pixel(x,y)=rgb(r,g,b)

```

Ver(index as integer) as double

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The vertical factors.
Notes:

Index from 0 to 19.
(Read and Write computed property)

2.2.2 Properties

DestinationPicture as Picture

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The destination picture.
Notes:

If you set this property, use a bitmap picture equal in size to the source picture.
If this property is nil, the Run method will create a picture.
(Read and Write property)

SourcePicture as Picture

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The source picture.
Notes:

Must be a bitmap picture.
(you can use the picture.BitmapMBS function for this)
(Read and Write property)

ValueCount as Integer

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The number of values set in the Hor and Ver array.

Example:

```
dim p as new PictureConvolutionMBS
```

```
p.Hor(0)=0.25  
p.Hor(1)=0.5  
p.Hor(2)=0.25  
p.ValueCount=3
```

Notes:

The index in the arrays goes from 0 to ValueCount-1.
Default is 3.
Use values like 1, 3, 5, 7, 9, 11, 13, 15, 17 or 19.
(Read and Write property)

2.3 class PictureEditorMBS

class PictureEditorMBS

Plugin Version: 11.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** class to edit picture data as a memoryblock in place.

Notes: This is the same code the plugin uses to edit pictures.

2.3.1 Methods

Data(Row as integer) as MemoryBlock

Plugin Version: 11.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The memoryblock with the original image data of the given row.

Notes:

Changes here will be visible in the picture.

This memoryblock has a size property with value 0!

No bound checking can be done by Realbasic on this memoryblock.

2.3.2 Properties

AllData as Memoryblock

Plugin Version: 11.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The memoryblock with the original image data.

Notes:

Changes here will be visible in the picture.

This memoryblock has a size property with value 0!

No bound checking can be done by Realbasic on this memoryblock.

(Read only property)

AllDataCopy as Memoryblock

Plugin Version: 11.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies the data for the current picture into a new memoryblock.

Notes:

Changes to this memoryblock will not be visible in the original picture.
(Read only property)

BlueOffset as Integer

Plugin Version: 11.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The platform dependent offset of the blue channel in the RGB data.

Notes:

A value between 0 and 3.
(Read only property)

BytesPerPixel as Integer

Plugin Version: 11.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Number of bytes per pixel.

Notes:

Most times 4, but for some platforms 3.
(Read only property)

DataPtr as Integer

Plugin Version: 11.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The memory address where the data is stored.

Notes:

Maybe useful for declares.
(Read only property)

GreenOffset as Integer

Plugin Version: 11.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The platform dependent offset of the green channel in the RGB data.

Notes:

A value between 0 and 3.
(Read only property)

Height as Integer

Plugin Version: 11.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The height of the image in pixels.

Notes: (Read only property)

Picture as Picture

Plugin Version: 11.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The original picture reference.

Notes: (Read only property)

RedOffset as Integer

Plugin Version: 11.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The platform dependent offset of the red channel in the RGB data.

Notes:

A value between 0 and 3.
(Read only property)

RowBytes as Integer

Plugin Version: 11.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The width of an image row in bytes.

Notes:

RowBytes can be width*bytesPerPixel, but often it is not.
(Read only property)

Width as Integer

Plugin Version: 11.3 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The width of the image in pixels.

Notes: (Read only property)

2.4 class PictureReaderMBS

class PictureReaderMBS

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A class to read picture data as a memoryblock.

Example:

```
dim pic as Picture = LogoMBS(500)
dim p as PictureReaderMBS
dim m as MemoryBlock
dim r,g,b,rRow,gRow,bRow,h1,w1,x,y,bpp as integer

// Create a new picture reader
p=NewPictureReaderMBS(pic)

h1=p.Height-1
w1=p.Width-1

bpp=p.BytesPerPixel
rRow=p.RedOffset
gRow=p.GreenOffset
bRow=p.BlueOffset
// in each row the red, blue and green channels have different offsets.
// but offsets are platform dependend

dim sum as Double

for y=0 to h1
// Get data in memory. This Memoryblock has a size property of 0!
m=p.Data(y)
```

```

r=rRow
g=gRow
b=bRow

for x=0 to w1

sum = sum + m.UInt8Value(r)
sum = sum + m.UInt8Value(g)
sum = sum + m.UInt8Value(b)

r=r+bpp
g=g+bpp
b=b+bpp
next

next

// show the sum of all pixels:
MsgBox "Sum with plugin is: "+str(sum)

// now try same in RB code:

dim surface as RGBSurface = pic.RGBSurface
dim c as color

sum = 0.0

for y=0 to h1
for x=0 to w1
c = surface.Pixel(x,y)

sum = sum + c.red
sum = sum + c.Green
sum = sum + c.Blue

next

next

surface = nil

MsgBox "Sum with RB Code is: "+str(sum)
quit

```

Notes: This is the same code the plugin uses to read pictures.

2.4.1 Methods

Data(Row as integer) as MemoryBlock

Plugin Version: 10.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The memoryblock with the original image data for this row.

Notes:

Changes here will be visible in the picture. (except for platforms where a copy is made of the data)

This memoryblock has a size property with value 0!

No bound checking can be done by Realbasic on this memoryblock.

See also:

- 2.4.2 Data as Memoryblock

87

2.4.2 Properties

BlueOffset as Integer

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The platform dependent offset of the blue channel in the RGB data.

Notes:

A value between 0 and 3.

(Read only property)

BytesPerPixel as Integer

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Number of bytes per pixel.

Notes:

Most times 4, but for some platforms 3.

(Read only property)

Data as Memoryblock

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The memoryblock with the original image data.

Notes:

Changes here will be visible in the picture. (except for platforms where a copy is made of the data)

This memoryblock has a size property with value 0!

No bound checking can be done by Realbasic on this memoryblock.

(Read only property)

See also:

- 2.4.1 Data(Row as integer) as MemoryBlock

86

DataCopy as Memoryblock

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies the data for the current picture into a new memoryblock.

Notes: (Read only property)

DataPtr as Integer

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The memory address where the data is stored.

Notes:

Maybe useful for declares.

(Read only property)

GreenOffset as Integer

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The platform dependend offset of the green channel in the RGB data.

Notes:

A value between 0 and 3.

(Read only property)

Height as Integer

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The height of the image in pixels.

Notes: (Read only property)

Picture as Picture

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The original picture reference.

Notes: (Read only property)

RedOffset as Integer

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The platform dependent offset of the red channel in the RGB data.

Notes:

A value between 0 and 3.
(Read only property)

RowBytes as Integer

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The width of an image row in bytes.

Notes:

RowBytes can be $\text{width} * \text{bytesPerPixel}$, but often it is not.
(Read only property)

Width as Integer

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The width of the image in pixels.

Notes: (Read only property)

2.5 class PictureWriterMBS

class PictureWriterMBS

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A class to build a picture by filling a memoryblock.

Example:

```

dim p as PictureWriterMBS
dim m as MemoryBlock
dim r,g,b,rRow,gRow,bRow,h1,w1,x,y,bpp as integer

// Create a new picture writer
p=NewPictureWriterMBS(512,512)

h1=p.Height-1
w1=p.Width-1

bpp=p.BytesPerPixel
rRow=p.RedOffset
gRow=p.GreenOffset
bRow=p.BlueOffset
// in each row the red, blue and green channels have different offsets.
// but offsets are platform dependend

for y=0 to h1
// Get data in memory. This Memoryblock has a size property of 0!
m=p.Data(y)
r=rRow
g=gRow
b=bRow

for x=0 to w1

m.UInt8Value(r)=x\2
m.UInt8Value(g)=y\2
m.UInt8Value(b)=x*y\2

r=r+bpp
g=g+bpp
b=b+bpp

```

```
next
```

```
next
```

```
// Use Render to make a picture object
dim pic as Picture = p.Render
Backdrop = pic
```

Notes: This is the same code the plugin uses to create pictures.

2.5.1 Methods

Data(Row as integer) as MemoryBlock

Plugin Version: 10.4 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The memoryblock with the original image data for the given row.

Notes:

Changes here will be visible in the picture.

This memoryblock has a size property with value 0!

No bound checking can be done by Realbasic on this memoryblock.

See also:

- 2.5.2 Data as Memoryblock

91

Render as picture

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates and returns a picture for this writer.

Notes:

The writer is destroyed with this call, so do not use it any more.
(one picture can be created with one writer currently)

2.5.2 Properties

BlueOffset as Integer

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The platform dependent offset of the blue channel in the RGB data.

Notes:

A value between 0 and 3.
(Read only property)

BytesPerPixel as Integer

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Number of bytes per pixel.

Notes:

Most times 4, but for some platforms 3.
(Read only property)

Data as Memoryblock

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The memoryblock with the original image data.

Notes:

Changes here will be visible in the picture.
This memoryblock has a size property with value 0!
No bound checking can be done by Realbasic on this memoryblock.
(Read only property)
See also:

- 2.5.1 Data(Row as integer) as MemoryBlock

DataCopy as Memoryblock

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Copies the data for the current picture into a new memoryblock.

Notes:

Changes to this memoryblock will not be visible in the rendered picture.
(Read only property)

DataPtr as Integer

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The memory address where the data is stored.

Notes:

Maybe useful for declares.
(Read only property)

GreenOffset as Integer

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The platform dependent offset of the green channel in the RGB data.

Notes:

A value between 0 and 3.
(Read only property)

Height as Integer

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The height of the image in pixels.

Notes: (Read only property)

Picture as Picture

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The buffer picture reference.

Notes:

If the writer uses a RB picture as buffer it is available here.
(depends on the actual implementation for a given platform whether this property is used)
(Read only property)

RedOffset as Integer

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The platform dependent offset of the red channel in the RGB data.

Notes:

A value between 0 and 3.
(Read only property)

RowBytes as Integer

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The width of an image row in bytes.

Notes:

RowBytes can be width*bytesPerPixel, but often it is not.
(Read only property)

Width as Integer

Plugin Version: 6.5 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The width of the image in pixels.

Notes: (Read only property)

2.6 class PictureSepiaMBS

class PictureSepiaMBS

Plugin Version: 4.1 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A class for doing a sepia effect.

2.6.1 Methods

close

Plugin Version: 4.1 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The destructor.
Notes:

There is no need to call this method except you want to free all resources of this object now without waiting for Realbasic to do it for you.

(e.g. some Realbasic versions crash on Windows if there are plugin objects not closed.)

Run as boolean

Plugin Version: 4.1 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Runs the picture effect.

Notes:

Fails if the pictures are not bitmap pictures. Source and Destination can be equal. If you provide a destination picture, the dimensions of source and destination must be equal.

For each pixel this method does:

```
sourcepixel=sourcepicture.pixel(x,y)
r=sourcepixel.red
g=sourcepixel.green
b=sourcepixel.blue
```

```
sum = r * RedFactor + g * GreenFactor + b * BlueFactor
```

```
r = sum + SepiaColor.red  
g = sum + SepiaColor.green  
b = sum + SepiaColor.blue
```

```
destinationpicture.pixel(x,y) = rgb(r,g,b)
```

2.6.2 Properties

DestinationPicture as Picture

Plugin Version: 4.1 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The destination picture.

Notes:

If you set this property, use a bitmap picture equal in size to the source picture.
If this property is nil, the Run method will create a picture.
(Read and Write property)

FactorBlue as Double

Plugin Version: 4.1 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The blue factor.

Notes: (Read and Write property)

FactorGreen as Double

Plugin Version: 4.1 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The green factor.

Notes: (Read and Write property)

FactorRed as Double

Plugin Version: 4.1 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The red factor.
Notes: (Read and Write property)

MaxX as Integer

Plugin Version: 4.1 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The maximal x coordinate to use.
Notes:

If 0 the width of the source picture defines this value.
(Read and Write property)

MaxY as Integer

Plugin Version: 4.1 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The maximal y coordinate to use.
Notes:

If 0 the height of the source picture defines this value.
(Read and Write property)

MinX as Integer

Plugin Version: 4.1 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The minimal x coordinate to use.
Notes:

Default is 0.
(Read and Write property)

MinY as Integer

Plugin Version: 4.1 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The minimal y coordinate to use.

Notes:

Default is 0.
(Read and Write property)

SepiaBlue as Integer

Plugin Version: 4.1 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The sepia color to use.

Notes:

Default is 0.
(Read and Write property)

SepiaGreen as Integer

Plugin Version: 4.1 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The sepia color to use.

Notes:

Default is 0.
(Read and Write property)

SepiaRed as Integer

Plugin Version: 4.1 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The sepia color to use.

Notes:

Default is 0.
(Read and Write property)

SourcePicture as Picture

Plugin Version: 4.1 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The source picture.

Notes:

Must be a bitmap picture.
 (you can use the picture.BitmapMBS function for this)
 (Read and Write property)

2.7 class PictureMatrixMBS**class PictureMatrixMBS**

Plugin Version: 4.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A class for matrix operations on a picture.

Notes: Can be used e.g. to sharpen a picture.

2.7.1 Methods**close**

Plugin Version: 4.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The destructor.

Notes:

There is no need to call this method except you want to free all resources of this object now without waiting for Realbasic to do it for you.

(e.g. some Realbasic versions crash on Windows if there are plugin objects not closed.)

Matrix(x as integer, y as integer) as integer

Plugin Version: 4.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The matrix used.

Notes:

X and Y are in range from 0 to 4.
 Values >255 are used for empty cells.

(Read and Write computed property)

Run as boolean

Plugin Version: 4.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Runs the process.
Notes: Returns true on success.

RunRGB(red as boolean, green as boolean, blue as boolean) as boolean

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Runs the process for the given channels.
Notes:

Returns true on success.
A few combinations are optimized for faster processing.
Still more optimization is possible.

2.7.2 Properties

DestinationPicture as Picture

Plugin Version: 4.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The destination picture.
Notes:

If this property is nil, a new picture will be placed here inside the Run method.
If you place a picture here, please use one created with newpicture with a 32bit depth.
(Read and Write property)

Displacement as Integer

Plugin Version: 4.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The displacement value.
Notes:

See the example project for details.
(Read and Write property)

MaxX as Integer

Plugin Version: 4.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The maximum x coordinate to use.

Notes:

Just for limiting the working area to a part of the picture.
(Read and Write property)

MaxY as Integer

Plugin Version: 4.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The maximum y coordinate to use.

Notes:

Just for limiting the working area to a part of the picture.
(Read and Write property)

MinX as Integer

Plugin Version: 4.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The minimum x coordinate to use.

Notes:

Just for limiting the working area to a part of the picture.
(Read and Write property)

MinY as Integer

Plugin Version: 4.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The minimum y coordinate to use.

Notes:

Just for limiting the working area to a part of the picture.
(Read and Write property)

ScaleFactor as Double

Plugin Version: 4.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A scaling factor.
Notes:

See the example project for details.
(Read and Write property)

SourcePicture as Picture

Plugin Version: 4.0 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The source picture.
Notes:

The run method will fail if this picture is not a 32bit deep picture created with newpicture.
(Read and Write property)

2.8 class PictureLut3DMBS

class PictureLut3DMBS

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A class for doing a LUT 3D on a picture.

2.8.1 Methods

close

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The destructor.
Notes:

There is no need to call this method except you want to free all resources of this object now without waiting for Realbasic to do it for you.

(e.g. some Realbasic versions crash on Windows if there are plugin objects not closed.)

Run as boolean

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Runs the picture effect.

Notes: Fails if the pictures are not bitmap pictures. Source and Destination can be equal. If you provide a destination picture, the dimensions of source and destination must be equal.

Table(r as integer, g as integer, b as integer, x as integer) as double

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The lut table.

Notes:

Indexes r, g and b go from 0 to 16 while x goes from 0 to 2.

(Read and Write computed property)

2.8.2 Properties

DestinationPicture as Picture

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The destination picture.

Notes:

If you set this property, use a bitmap picture equal in size to the source picture.

If this property is nil, the Run method will create a picture.

(Read and Write property)

MaxX as Integer

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The maximal x coordinate to use.

Notes:

If 0 the width of the source picture defines this value.
(Read and Write property)

MaxY as Integer

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The maximal y coordinate to use.

Notes:

If 0 the height of the source picture defines this value.
(Read and Write property)

MinX as Integer

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The minimal x coordinate to use.

Notes:

Default is 0.
(Read and Write property)

MinY as Integer

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The minimal y coordinate to use.

Notes:

Default is 0.
(Read and Write property)

SourcePicture as Picture

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The source picture.
Notes:

Must be a bitmap picture.
 (you can use the picture.BitmapMBS function for this)
 (Read and Write property)

2.9 class PictureMatrix3DMBS**class PictureMatrix3DMBS**

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A class for doing a 3D picture matrix.

2.9.1 Methods**close**

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The destructor.
Notes:

There is no need to call this method except you want to free all resources of this object now without waiting for Realbasic to do it for you.
 (e.g. some Realbasic versions crash on Windows if there are plugin objects not closed.)

Matrix(x as integer, y as integer) as double

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The matrix to use.
Notes:

The indexes x and y are 0 based.
 (Read and Write computed property)

Run as boolean

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Runs the picture effect.

Notes: Fails if the pictures are not bitmap pictures. Source and Destination can be equal. If you provide a destination picture, the dimensions of source and destination must be equal.

2.9.2 Properties

DestinationPicture as Picture

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The destination picture.

Notes:

If you set this property, use a bitmap picture equal in size to the source picture.

If this property is nil, the Run method will create a picture.

(Read and Write property)

MaxX as Integer

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The maximal x coordinate to use.

Notes:

If 0 the width of the source picture defines this value.

(Read and Write property)

MaxY as Integer

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The maximal y coordinate to use.

Notes:

If 0 the height of the source picture defines this value.

(Read and Write property)

MinX as Integer

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The minimal x coordinate to use.

Notes:

Default is 0.
(Read and Write property)

MinY as Integer

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The minimal y coordinate to use.

Notes:

Default is 0.
(Read and Write property)

SourcePicture as Picture

Plugin Version: 4.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The source picture.

Notes:

Must be a bitmap picture.
(you can use the picture.BitmapMBS function for this)
(Read and Write property)

2.10 class PictureMinMaxMBS

class PictureMinMaxMBS

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A class to find the minimum/maximum pixel values.

Example:

```
dim pic as Picture = LogoMBS(500)
dim m as new PictureMinMaxMBS

if m.FindAll(pic) then
break // check values in debugger
end if
```

Notes:

This class offers several Find functions.

Please choose carefully which one you use as it's faster to use e.g. FindRed instead of FindMinRed and FindMaxRed together.

2.10.1 Methods

FindAll(p as picture) as boolean

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Searches for the minimum and maximum pixels.

Notes: Sets all fields.

FindBlue(p as picture) as boolean

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Searches for the minimum and maximum blue pixels.

Notes: Sets BlueMaxX, BlueMax, BlueMinX, BlueMinY, BlueMin and BlueMaxY.

FindGreen(p as picture) as boolean

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Searches for the minimum and maximum green pixels.

Notes: Sets GreenMaxX, GreenMax, GreenMinX, GreenMinY, GreenMin and GreenMaxY.

FindMaxAll(p as picture) as boolean

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Searches for the maximum pixels.

Notes: Sets RedMaxX, RedMax, RedMaxY, GreenMaxX, GreenMax, GreenMaxY, BlueMaxX, BlueMax and BlueMaxY.

FindMaxBlue(p as picture) as boolean

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Searches for the maximum blue pixel.

Notes: Sets BlueMaxX, BlueMax and BlueMaxY.

FindMaxGreen(p as picture) as boolean

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Searches for the maximum green pixel.

Notes: Sets GreenMaxX, GreenMax and GreenMaxY.

FindMaxRed(p as picture) as boolean

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Searches for the maximum red pixel.

Notes: Sets RedMaxX, RedMax and RedMaxY.

FindMaxSum(p as picture) as boolean

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Searches for the maximum sum pixel.

Notes:

The sum of a pixel is the sum of all color channels of this pixel (red+Sum+blue).
Sets SumMaxX, SumMax and SumMaxY.

FindMinAll(p as picture) as boolean

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Searches for the minimum pixels.

Notes: Sets RedMinX, RedMin, RedMinY, GreenMinX, GreenMin, GreenMinY, BlueMinX, BlueMin and BlueMinY.

FindMinBlue(p as picture) as boolean

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Searches for the minimum blue pixel.

Notes: Sets BlueMinX, BlueMin and BlueMinY.

FindMinGreen(p as picture) as boolean

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Searches for the minimum green pixel.

Notes: Sets GreenMinX, GreenMin and GreenMinY.

FindMinRed(p as picture) as boolean

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Searches for the minimum red pixel.

Notes: Sets RedMinX, RedMin and RedMinY.

FindMinSum(p as picture) as boolean

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Searches for the minimum sum pixel.

Notes:

The sum of a pixel is the sum of all color channels of this pixel (red+green+blue).
Sets SumMinX, SumMin and SumMinY.

FindRed(p as picture) as boolean

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Searches for the minimum and maximum red pixels.

Notes: Sets RedMaxX, RedMax, RedMinX, RedMinY, RedMin and RedMaxY.

FindSum(p as picture) as boolean

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Searches for the minimum and maximum sum pixels.

Notes:

The sum of a pixel is the sum of all color channels of this pixel (red+Sum+blue).
Sets SumMaxX, SumMax, SumMinX, SumMinY, SumMin and SumMaxY.

2.10.2 Properties

BlueMax as Integer

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The maximum blue color value.

Notes:

Range: 0 to 255.
(Read and Write property)

BlueMaxX as Integer

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The location of the pixel with the maximum blue value.

Notes:

Range: 0 to Picture.Width-1. Set to -1 on any error.
(Read and Write property)

BlueMaxY as Integer

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The location of the pixel with the maximum blue value.

Notes:

Range: 0 to Picture.Height-1. Set to -1 on any error.
(Read and Write property)

BlueMin as Integer

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The minimum blue color value.

Notes:

Range: 0 to 255.
(Read and Write property)

BlueMinX as Integer

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The location of the pixel with the minimum blue value.

Notes:

Range: 0 to Picture.Width-1. Set to -1 on any error.
(Read and Write property)

BlueMinY as Integer

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The location of the pixel with the minimum blue value.

Notes:

Range: 0 to Picture.Height-1. Set to -1 on any error.
(Read and Write property)

GreenMax as Integer

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The maximum green color value.

Notes:

Range: 0 to 255
(Read and Write property)

GreenMaxX as Integer

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The location of the pixel with the maximum green value.

Notes:

Range: 0 to Picture.Width-1. Set to -1 on any error.
(Read and Write property)

GreenMaxY as Integer

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The location of the pixel with the maximum green value.

Notes:

Range: 0 to Picture.Height-1. Set to -1 on any error.
(Read and Write property)

GreenMin as Integer

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The minimum green color value.

Notes:

Range: 0 to 255
(Read and Write property)

GreenMinX as Integer

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The location of the pixel with the minimum green value.

Notes:

Range: 0 to Picture.Width-1. Set to -1 on any error.
(Read and Write property)

GreenMinY as Integer

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The location of the pixel with the minimum green value.

Notes:

Range: 0 to Picture.Height-1. Set to -1 on any error.
(Read and Write property)

RedMax as Integer

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The maximum red color value.

Notes:

Range: 0 to 255
(Read and Write property)

RedMaxX as Integer

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The location of the pixel with the maximum red value.

Notes:

Range: 0 to Picture.Width-1. Set to -1 on any error.
(Read and Write property)

RedMaxY as Integer

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The location of the pixel with the maximum red value.

Notes:

Range: 0 to Picture.Height-1. Set to -1 on any error.
(Read and Write property)

RedMin as Integer

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The minimum red color value.

Notes:

Range: 0 to 255
(Read and Write property)

RedMinX as Integer

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The location of the pixel with the minimum red value.

Notes:

Range: 0 to Picture.Width-1. Set to -1 on any error.
(Read and Write property)

RedMinY as Integer

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The location of the pixel with the minimum red value.

Notes:

Range: 0 to Picture.Height-1. Set to -1 on any error.
(Read and Write property)

SumMax as Integer

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The maximum sum color value.

Notes:

sum=red+blue+green

Range: 0 to 765
(Read and Write property)

SumMaxX as Integer

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The location of the pixel with the maximum sum value.

Notes:

Range: 0 to Picture.Width-1. Set to -1 on any error.
(Read and Write property)

SumMaxY as Integer

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The location of the pixel with the maximum sum value.

Notes:

Range: 0 to Picture.Height-1. Set to -1 on any error.
(Read and Write property)

SumMin as Integer

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The minimum sum color value.

Notes:

sum=red+blue+green

Range: 0 to 765

(Read and Write property)

SumMinX as Integer

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The location of the pixel with the minimum sum value.

Notes:

Range: 0 to Picture.Width-1. Set to -1 on any error.

(Read and Write property)

SumMinY as Integer

Plugin Version: 3.4 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The location of the pixel with the minimum sum value.

Notes:

Range: 0 to Picture.Height-1. Set to -1 on any error.

(Read and Write property)

2.11 class BarcodeScannerMBS

class BarcodeScannerMBS

Plugin Version: 8.0 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A class to read a barcode from a given picture.

2.11.1 Methods

Scan(p as picture) as boolean

Plugin Version: 8.0 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Scans for a barcode on the picture.

Notes:

The barcode is searched on the middle vertical line from the left to right.

So the picture you pass can be as small as just one pixel height.
The barcode should be horizontal centered in that picture for best results.

Returns true on success and false on failure.

See also:

- 2.11.1 Scan(p as picture, lines() as integer) as boolean

117

Scan(p as picture, lines() as integer) as boolean

Plugin Version: 8.0 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Scans for a barcode on the picture.

Example:

```
dim i as integer
dim lines(-1) as integer
dim b as BarcodeScannerMBS
dim p as Picture
// set b to your scanner and p to your picture

for i=0 to 99
lines.append i*10 // search every 10th line on a 1000 pixel high image.
next

if b.scan(p,lines) then
// ok
end if
```

Notes:

The barcode is searched on the lines with the given offsets from the left to right.

The lines array must have at least one entry specifying the lines to search on.
If the values in the lines array are out of bounds, they are ignored. The first line has the value 0.

So the picture you pass can be as small as just one pixel height.
The barcode should be horizontal centered in that picture for best results.

Returns true on success and false on failure.
See also:

- 2.11.1 Scan(p as picture) as boolean

117

2.11.2 Properties

Barcode as String

Plugin Version: 8.0 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The barcode result of the last scan.

Notes:

If the last scan was successful, this property has a value.
If the last scan failed, this property is empty.
(Read and Write property)

CheckDigits as Boolean

Plugin Version: 8.0 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Whether to calculate the checksum.

Notes:

Normal 12 or 13 digit barcodes have the last number being a checksum.
If it does not match, the barcode is declined.

Default is false.
(Read and Write property)

LastBarcode as String

Plugin Version: 8.0 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The barcode result of the last successful scan.

Notes:

If a scan fails, this value still has the value of the last successful scan.
(Read and Write property)

LastPicture as Picture

Plugin Version: 8.0 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The last picture used for the scan.

Notes: (Read and Write property)

MinimumLength as Integer

Plugin Version: 8.0 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The minimum length of barcodes.

Notes:

Set to 0 to disable.

To avoid false barcodes, any barcode is rejected which does not have the sufficient length.

Default is 13.

(Read and Write property)

Mode as Integer

Plugin Version: 8.0 Console & Web: No Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The mode of the scanner.

Notes:

Mode 0 is to scan EANs.

Mode 1 is to scan 2/5 family barcodes.

(Read and Write property)

2.12 class PaletteMBS

class PaletteMBS

Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** A color PaletteMBS.

2.12.1 Methods

Col(i as integer) as color

Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The color array.

Notes:

Index goes from 0 to count-1.

(Read and Write computed property)

CountColors as integer

Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Counts how many different colors are in the palette.

Mem as memoryblock

Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns a memoryblock to the binary data of this PaletteMBS.

Notes:

The block is 1024 bytes big and contains 256 32bit values like 00RRGGBB.
The memoryblock is only valid as long as the PaletteMBS object is living.

NewPicture(width as integer,height as integer) as picture

Console & Web: No Mac: Yes, Win: Yes, Linux: No, . **Function:** Creates a new picture using this PaletteMBS.

Notes: On Windows, RB can only work with 32bit picture so this function will return a 32bit picture for you.

2.12.2 Properties

count as integer

Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** How many colors are inside this PaletteMBS.

Notes:

This property should be 2, 4, 16 or 256.

Default is 256.

(Read and Write property)

2.13 class PaletteCalculatorMBS

class PaletteCalculatorMBS

Plugin Version: 8.7 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** This class allows you to calculate an 8 bit image from a RGB image and back.

Notes:

You can create the best matching palette for a given image.

If you have several images which should share the same palette, you can draw them first on one big picture before calculating the

2.13.1 Methods

Col(*i as integer*) as color

Plugin Version: 8.7 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** The color array.
Notes:

Index goes from 0 to count-1.
 (Read and Write computed property)

CountColors as integer

Plugin Version: 8.7 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Counts how many different colors are in the palette.

CreatePicturePalette(*Pic as picture*) as integer

Plugin Version: 8.7 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a picture palette based on the picture.
Notes: This function checks which colors are very often used in the image and builds a palette which may be better for this image than the default system palette.

GetIndexOfColor(*col as color*) as integer

Plugin Version: 8.7 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Searches the index in the palette for the given color.
Notes: Returns -1 if the color is not found.
 See also:

- 2.13.1 GetIndexOfColor(*r as integer, g as integer, b as integer*) as integer

GetIndexOfColor(r as integer, g as integer, b as integer) as integer

Plugin Version: 8.7 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Searches the index in the palette for the given color.

Notes: Returns -1 if the color is not found.

See also:

- 2.13.1 GetIndexOfColor(col as color) as integer 122

GetNearestIndexOfColor(col as color) as integer

Plugin Version: 8.7 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Searches the index in the palette for the best matching color.

Notes:

The best color is the color with the lowest value:

$$\text{value} = (\text{r-col}(\text{index}).\text{red})^2 + (\text{g-col}(\text{index}).\text{green})^2 + (\text{b-col}(\text{index}).\text{blue})^2$$

Returns -1 if the color is not found (should never happen).

See also:

- 2.13.1 GetNearestIndexOfColor(r as integer, g as integer, b as integer) as integer 123

GetNearestIndexOfColor(r as integer, g as integer, b as integer) as integer

Plugin Version: 8.7 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Searches the index in the palette for the best matching color.

Notes:

The best color is the color with the lowest value:

$$\text{value} = (\text{r-col}(\text{index}).\text{red})^2 + (\text{g-col}(\text{index}).\text{green})^2 + (\text{b-col}(\text{index}).\text{blue})^2$$

Returns -1 if the color is not found (should never happen).

See also:

- 2.13.1 GetNearestIndexOfColor(col as color) as integer 123

Transform(mem as memoryblock, width as integer, height as integer) as picture

Plugin Version: 8.7 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Transforms a 8 bit picture to a RGB picture.

Notes:

The memoryblock must have the 8 bit picture data inside with each row being width bytes big. The memoryblock must have at least width*height bytes.

Returns nil on any error.

See also:

- 2.13.1 Transform(Pic as picture) as memoryblock 124

Transform(Pic as picture) as memoryblock

Plugin Version: 8.7 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a memoryblock with 8 bit picture data.

Notes:

The resulting memoryblock has width*height bytes.

Each RGB color in the picture is looked up in the palette and used to fill the memoryblock.

See also:

- 2.13.1 Transform(mem as memoryblock, width as integer, height as integer) as picture 124

TransformBetterDithering(Pic as picture) as memoryblock

Plugin Version: 8.7 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a memoryblock with 8 bit picture data.

Notes:

The resulting memoryblock has width*height bytes.

Each RGB color in the picture is looked up in the palette and used to fill the memoryblock.

This method uses dithering to make the picture looking better than with a better transform using code like Floyd-Steinberg.

TransformFastDithering(Pic as picture) as memoryblock

Plugin Version: 8.7 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a memoryblock with 8 bit picture data.

Notes:

The resulting memoryblock has width*height bytes.

Each RGB color in the picture is looked up in the palette and used to fill the memoryblock.

This method uses dithering to make the picture looking better than with a simple transform.

2.13.2 Properties

Count as Integer

Plugin Version: 8.7 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** How many colors are inside this PaletteMBS.

Notes:

This property should be 2, 4, 16 or 256.

Default is 256.

(Read and Write property)

Chapter 3

Icon Service

3.1 Globals

CompositeIconsMBS(ForeGround as IconMBS, BackGround as IconMBS) as IconMBS

Plugin Version: 5.1 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** Combines two icons.
Example:

```
dim i as IconMBS // global

Sub Open()
dim a,b as IconMBS

b=new IconMBS(SpecialFolder.Desktop)
a=new IconMBS(app.ApplicationFileMBS)

i=CompositeIconsMBS(a,b)

End Sub

Sub Paint(g As Graphics)
i.DrawIcon(g,0,0,128,128)
End Sub
```

Notes: Returns nil on any error (e.g. one of the two icons is invalid or nil).

NewIconFamilyMBS as IconFamilyMBS

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** Creates a new empty IconFamily object.

Example:

```
dim i as IconFamilyMBS = NewIconFamilyMBS
```

Notes: Returns nil on any error.

NewIconFamilyMBSFromScrap as IconFamilyMBS

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** Creates a new IconFamily object with the data of a "icns" resource from the clipboard.

Example:

```
// put the Finder Icon on the clipboard

dim i as new IconMBS("FNDR", "MACS")
i.IconFamily.PutOnScrap

// and get it back

dim n as IconFamilyMBS = NewIconFamilyMBSFromScrap
Backdrop = n.Thumbnail32BitData
```

Notes: Returns nil on any error.

3.2 class IconFamilyMBS**class IconFamilyMBS**

Plugin Version: 2.6 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** A class for an icon family on Mac OS.

3.2.1 Methods

close

Plugin Version: 4.1 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** The destructor.
Notes:

There is no need to call this method except you want to free all resources of this object now without waiting for Realbasic to do it for you.
 (e.g. some Realbasic versions crash on Windows if there are plugin objects not closed.)

Data as string

Plugin Version: 5.3 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** The data of this icon in the ICNS format.

Example:

```
dim g as FolderItem
dim i as IconMBS
dim f as IconFamilyMBS
dim s as string
dim b as BinaryStream

g=SpecialFolder.Desktop
i=new IconMBS(g) // get icon from desktop folder on Mac OS X
f=i.IconFamily
Backdrop=f.Thumbnail32BitData

s=f.Data

MsgBox str(lenb(s))+” bytes”

g=SpecialFolder.Desktop.Child(”Desktop folder icon.icns”)
b=g.CreateBinaryFile(”Icon”) // you need to define this type!
b.write s
b.close

g.launch // shows in preview the icns file
```

Notes:

Returns "" on low memory or any error.
(Read and Write computed property)

GetIconImage(size as integer, byref pic as picture, byref mask as picture) as boolean

Plugin Version: 7.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** Returns an icon with the given size.

Example:

```
dim p,m as Picture
dim s as IconFamilyMBS // your icon

if s.GetIconImage(512,p,m) then
window1.Backdrop=p
window2.Backdrop=m
end if
```

Notes:

Size may be 16, 32, 48, 128, 256 or 512.
Returns true on success and false on failure.
Works only on Mac OS X 10.5

Lasterror is -50 if the given size is not available.

Huge1BitData as picture

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** If such an icon is included in this icon family, this function returns it.

Example:

```
dim i as new IconMBS("FNDR", "MACS")
Backdrop = i.IconFamily.Huge1BitData
```

Notes:

Lasterror is set.
(Read and Write computed property)

Huge1BitMask as picture

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** If such an icon is included in this icon family, this function returns it.

Example:

```
dim i as new IconMBS("FNDR", "MACS")  
Backdrop = i.IconFamily.Huge1BitMask
```

Notes:

Lasterror is set.
(Read and Write computed property)

Huge32BitData as picture

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** If such an icon is included in this icon family, this function returns it.

Example:

```
dim i as new IconMBS("FNDR", "MACS")  
Backdrop = i.IconFamily.Huge32BitData
```

Notes:

Lasterror is set.
(Read and Write computed property)

Huge4BitData as picture

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** If such an icon is included in this icon family, this function returns it.

Example:

```
dim i as new IconMBS("FNDR", "MACS")
Backdrop = i.IconFamily.Huge4BitData
```

Notes:

Lasterror is set.
(Read and Write computed property)

Huge8BitData as picture

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** If such an icon is included in this icon family, this function returns it.

Example:

```
dim i as new IconMBS("FNDR", "MACS")
Backdrop = i.IconFamily.Huge8BitData
```

Notes:

Lasterror is set.
(Read and Write computed property)

Huge8BitMask as picture

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** If such an icon is included in this icon family, this function returns it.

Example:

```
dim i as new IconMBS("FNDR", "MACS")
Backdrop = i.IconFamily.Huge8BitMask
```

Notes:

Lasterror is set.
(Read and Write computed property)

Large1BitData as picture

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** If such an icon is included in this icon family, this function returns it.

Example:

```
dim i as new IconMBS("FNDR", "MACS")  
Backdrop = i.IconFamily.Large1BitData
```

Notes:

Lasterror is set.
(Read and Write computed property)

Large1BitMask as picture

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** If such an icon is included in this icon family, this function returns it.

Example:

```
dim i as new IconMBS("FNDR", "MACS")  
Backdrop = i.IconFamily.Large1BitMask
```

Notes:

Lasterror is set.
(Read and Write computed property)

Large32BitData as picture

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** If such an icon is included in this icon family, this function returns it.

Example:

```
dim i as new IconMBS("FNDR", "MACS")
Backdrop = i.IconFamily.Large32BitData
```

Notes:

Lasterror is set.
(Read and Write computed property)

Large4BitData as picture

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** If such an icon is included in this icon family, this function returns it.

Example:

```
dim i as new IconMBS("FNDR", "MACS")
Backdrop = i.IconFamily.Large4BitData
```

Notes:

Lasterror is set.
(Read and Write computed property)

Large8BitData as picture

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** If such an icon is included in this icon family, this function returns it.

Example:

```
dim i as new IconMBS("FNDR", "MACS")
Backdrop = i.IconFamily.Large8BitData
```

Notes:

Lasterror is set.
(Read and Write computed property)

Large8BitMask as picture

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** If such an icon is included in this icon family, this function returns it.

Example:

```
dim i as new IconMBS("FNDR", "MACS")  
Backdrop = i.IconFamily.Large8BitMask
```

Notes:

Lasterror is set.
(Read and Write computed property)

PutOnScrap

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** Puts the Icon as an "icns" resource on the clipboard.

Example:

```
dim i as new IconMBS("FNDR", "MACS")  
i.IconFamily.PutOnScrap
```

Notes: LastError is set.

Register(creator as string, type as string) as IconMBS

Plugin Version: 4.3 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** Registers an icon.

Notes:

The current icon stored in the iconfamily is saved in the global icon list with the given type and creator combination.

On success the new IconMBS object is returned. In case the IconMBS object is destroyed, the icon will automatically be removed from the icon list.

Lasterror is set.

SetIconImage(pic as picture, mask as picture) as boolean

Plugin Version: 7.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** Sets the icon data.

Example:

```
dim p as Picture // your picture
dim m as Picture // the mask for the picture
dim i as IconFamilyMBS // your icon family

// 512, 256, 128 Pixel images for Leopard
dim ps,ms as Picture
ps=p.ScaleMBS(512,512)
ms=m.ScaleMBS(512,512)
call i.SetIconImage(ps,ms)
ps=p.ScaleMBS(512,256)
ms=m.ScaleMBS(512,256)
call i.SetIconImage(ps,ms)
ps=p.ScaleMBS(512,128)
ms=m.ScaleMBS(512,128)
call i.SetIconImage(ps,ms)
```

Notes:

Size of the pictures may be 16, 32, 48, 128, 256 or 512.

pic and mask must not be nil.

pic.width, mask.width, pic.height and mask.height must all be same.

Returns true on success and false on failure.

Works only on Mac OS X 10.5

Small1BitData as picture

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** If such an icon is included in this icon family, this function returns it.

Example:

```
dim i as new IconMBS("FNDR", "MACS")  
Backdrop = i.IconFamily.Small1BitData
```

Notes:

Lasterror is set.
(Read and Write computed property)

Small1BitMask as picture

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** If such an icon is included in this icon family, this function returns it.

Example:

```
dim i as new IconMBS("FNDR", "MACS")  
Backdrop = i.IconFamily.Small1BitMask
```

Notes:

Lasterror is set.
(Read and Write computed property)

Small32BitData as picture

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** If such an icon is included in this icon family, this function returns it.

Example:

```
dim i as new IconMBS("FNDR", "MACS")
```

```
Backdrop = i.IconFamily.Small32BitData
```

Notes:

Lasterror is set.
(Read and Write computed property)

Small4BitData as picture

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** If such an icon is included in this icon family, this function returns it.

Example:

```
dim i as new IconMBS("FNDR", "MACS")  
Backdrop = i.IconFamily.Small4BitData
```

Notes:

Lasterror is set.
(Read and Write computed property)

Small8BitData as picture

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** If such an icon is included in this icon family, this function returns it.

Example:

```
dim i as new IconMBS("FNDR", "MACS")  
  
Backdrop = i.IconFamily.Small8BitData
```

Notes:

Lasterror is set.

(Read and Write computed property)

Small8BitMask as picture

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** If such an icon is included in this icon family, this function returns it.

Example:

```
dim i as new IconMBS("FNDR", "MACS")
```

```
Backdrop = i.IconFamily.Small8BitMask
```

Notes:

Lasterror is set.

(Read and Write computed property)

Thumbnail32BitData as picture

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** If such an icon is included in this icon family, this function returns it.

Example:

```
// Example about a function to return a REALbasic picture with an icon and its
// mask.
```

```
//
```

```
Function geticonpicture As picture
```

```
// You have to add more support, like for example for Huge and Small icons.
```

```
dim icon as IconMBS
```

```
dim p as picture
```

```
dim b as picture
```

```
dim m as picture
```

```
icon=new IconMBS("APPL","sbkt")
```

```
if icon.valid then
```

```
dim i as IconFamilyMBS = icon.IconFamily
```

```
p=newpicture(128,128,32)

b=i.Thumbnail32BitData
if b<>nil then
p.graphics.drawpicture b,0,0
p.mask.graphics.drawpicture i.thumbnail8BitMask,0,0
return p
end if

m=i.Large1BitMask
if m<>nil then
b=i.Large32BitData
if B=nil then
b=i.Large8BitData
end if
if b=nil then
b=i.Large4BitData
end if
if b=nil then
b=i.large1BitData
end if

p.graphics.drawpicture b,0,0,128,128,0,0,32,32
b=i.large8BitMask
if b<>nil then
m=b
end if
p.mask.graphics.drawpicture m,0,0,128,128,0,0,32,32
return p
end if

End Function
```

Notes:

Lasterror is set.
(Read and Write computed property)

Thumbnail8BitMask as picture

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** If such an icon is included in this icon family, this function returns it.

Example:

```
dim i as new IconMBS("FNDR", "MACS")

Backdrop = i.IconFamily.Thumbnail8BitMask
```

Notes:

Lasterror is set.
(Read and Write computed property)

WriteFile(f as folderitem)

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** Writes the icon family into an icon file.

Example:

```
dim pict,mask as Picture
dim iconfamily as IconFamilyMBS
dim f as FolderItem

// create pictures
pict=NewPicture(128,128,32)
mask=NewPicture(128,128,32)

pict.Graphics.ForeColor=rgb(255,0,0)
pict.Graphics.FillOval 0,0,128,128

mask.Graphics.ForeColor=rgb(0,0,0)
mask.Graphics.FillOval 0,0,128,128

// make an icon family
iconfamily=NewIconFamilyMBS
iconfamily.Thumbnail32BitData=pict
iconfamily.Thumbnail8BitMask=mask
' you may fill more like iconfamily.Large32BitData...
```

```
f=SpecialFolder.Desktop.Child("test.icns")

// Save *.ICNS file:
iconfamily.WriteFile f

if f.AddCustomIconMBS(iconfamily,false)=0 then
// succesfully added custom icon
end if
```

Notes: LastError is set.

3.2.2 Properties

Dither as boolean

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: Yes, Linux: No, . **Function:** whether on setting an icon the picture is dithered to the new color depth.

Notes:

Dithered pictures look normally better.
(Read and Write property)

Handle as integer

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** The Handle for this Icon family.

Notes:

Value is a IconFamilyHandle.
(Read and Write property)

LastError as integer

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** The last error code.

Notes:

The last function was successful if lasterror is 0.
 If the last function was not available on this machine, the value is set to -1.
 Other values are Mac OS error codes.
 (Read and Write property)

Release as boolean

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: Yes, Linux: No, . **Function:** whether to release the handle in the destructor.
Notes: (Read and Write property)

Valid as boolean

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** whether the icon family is valid.
Notes: (Read and Write property)

3.3 class IconMBS

class IconMBS

Plugin Version: 2.6 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** A class for an icon on Mac OS.

Example:

// A function which will try to return an icon for the given type/creator including the Mask.

```
Function GetIconPicture(macCreator as string, macType as string, size as integer) As picture
dim icn as IconMBS
dim icf as IconFamilyMBS
dim pic, tmp as Picture

icn = new IconMBS(macType, macCreator)

if icn<>nil and icn.valid then
icf = icn.IconFamily
end if
```

```
if icf<>nil and icf.Valid then
pic = NewPicture(size, size, 32)

// Try Thumbnail
if size>32 then
tmp = icf.Thumbnail32BitData
if tmp<>nil then
pic.Graphics.DrawPicture tmp, 0, 0, size, size, 0, 0, tmp.width,tmp.Height

tmp = icf.Thumbnail8BitMask
if tmp<>nil then
pic.Mask.Graphics.DrawPicture tmp, 0, 0, size, size, 0, 0, tmp.width, tmp.Height
end if
Return pic
end if
end if

// Try Large Icon in 32 bit
tmp = icf.Large32BitData
if tmp<>nil then
pic.Graphics.DrawPicture tmp, 0, 0, size, size, 0, 0, tmp.width,tmp.Height

tmp = icf.Large8BitMask
if tmp<>nil then
pic.Mask.Graphics.DrawPicture tmp, 0, 0, size, size, 0, 0, tmp.width, tmp.Height
end if
Return pic
end if

// Try Large Icon in 8 bit
tmp = icf.Large8BitData
if tmp<>nil then
pic.Graphics.DrawPicture tmp, 0, 0, size, size, 0, 0, tmp.width,tmp.Height

tmp = icf.Large1BitMask
if tmp<>nil then
pic.Mask.Graphics.DrawPicture tmp, 0, 0, size, size, 0, 0, tmp.width, tmp.Height
end if
return pic
end if

// You may add more like e.g. Small or Huge Icons
end if
```

Exception // on any error, just return nil

```
// Call like:
// Backdrop=GetIconPicture("R*ch","TEXT",128)
End Function
```

3.3.1 Methods

Constructor(f as folderitem)

Plugin Version: 2.6 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** Loads the icon for this file/folder/volume.

Example:

```
// in a paint event:
dim i as new IconMBS(SpecialFolder.Desktop)
i.DrawIcon(g, 0, 0, 128, 128)
```

Notes:

The example "GetIcon.rb" shows how to get the file icons.
A custom icon is preferred (ID -16455).
See also:

- 3.3.1 Constructor(type as string, creator as string) 145
- 3.3.1 Constructor(type as string, creator as string, extension as string, mime as string) 146

Constructor(type as string, creator as string)

Plugin Version: 2.6 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** Loads the icon for this type and creator code combination.

Example:

```
// in a paint event:
dim i as new IconMBS("FNDR", "MACS")
```

```
i.DrawIcon(g, 0, 0, 128, 128)
```

Notes: The example "GetIcon.rb" shows how to get the predefined icons from the system.
See also:

- 3.3.1 Constructor(f as folderitem) 145
- 3.3.1 Constructor(type as string, creator as string, extension as string, mime as string) 146

Constructor(type as string, creator as string, extension as string, mime as string)

Plugin Version: 9.2 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** Loads the icon base on the given information.

Example:

```
Sub Paint(g As Graphics)
```

```
// in a window paint event:
```

```
dim i as IconMBS
```

```
dim type, creator, extension, mime as string
```

```
type=""
```

```
creator=""
```

```
extension="jpg"
```

```
mime=""
```

```
i=new iconmbs(type, creator, extension, mime)
```

```
// draws jpeg icon
```

```
i.DrawIcon(g,0,0,128,128)
```

```
type=""
```

```
creator=""
```

```
extension=""
```

```
mime="video/quicktime"
```

```
i=new iconmbs(type, creator, extension, mime)
```

```
// draws quicktime movie icon
```

```
i.DrawIcon(g,128,0,128,128)
```

```
type="TEXT"
```

```
creator="MSWD"
```

```
extension=""
```

```

mime=""

i=new iconmbs(type, creator, extension, mime)
// draws microsoft word text file icon
i.DrawIcon(g,0,128,128,128)

type=""
creator="GKON"
extension="jpg"
mime=""

i=new iconmbs(type, creator, extension, mime)
// draws graphic converter jpeg file icon
i.DrawIcon(g,128,128,128,128)

```

End Sub

Notes:

All parameters can be empty strings if you don't know this information.

Requires Mac OS X 10.3 to work properly.

See also:

- 3.3.1 Constructor(f as folderitem) 145
- 3.3.1 Constructor(type as string, creator as string) 145

DrawIcon(g as graphics,x as integer,y as integer,width as integer,height as integer)

Plugin Version: 2.6 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** Draws the icon.

Example:

```

// in a paint event:

dim i as new IconMBS("FNDR", "MACS")

i.DrawIcon(g, 0, 0, 128, 128)

```

Notes:

DrawIcon with align and transform set to none.

Not supported for Cocoa. Please use DrawIconCGContext there.

See also:

- 3.3.1 DrawIcon(g as graphics,x as integer,y as integer,width as integer,height as integer,align as integer) 148
- 3.3.1 DrawIcon(g as graphics,x as integer,y as integer,width as integer,height as integer,align as integer,transform as integer) 148

DrawIcon(g as graphics,x as integer,y as integer,width as integer,height as integer,align as integer)

Plugin Version: 2.6 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** Draws the icon.

Example:

// in a paint event:

```
dim i as new IconMBS("FNDR", "MACS")
```

```
i.DrawIcon(g, 0, 0, 128, 128, 8)
```

Notes:

DrawIcon with transform set to none.

Not supported for Cocoa. Please use DrawIconCGContext there.

See also:

- 3.3.1 DrawIcon(g as graphics,x as integer,y as integer,width as integer,height as integer) 147
- 3.3.1 DrawIcon(g as graphics,x as integer,y as integer,width as integer,height as integer,align as integer,transform as integer) 148

DrawIcon(g as graphics,x as integer,y as integer,width as integer,height as integer,align as integer,transform as integer)

Plugin Version: 2.6 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** Draws the icon.

Example:

```
// in a paint event:
dim i as new IconMBS("FNDR", "MACS")
i.DrawIcon(g, 0, 0, 128, 128, 0, & h4000)
```

Notes:

Align and Transform are optional parameters.

The coordinates inside the graphics objects are absolute to the picture or window where the graphics object came from.

Align constants:

None	0
VerticalCenter	1
Top	2
Bottom	3
HorizontalCenter	4
AbsoluteCenter	5
CenterTop	6
CenterBottom	7
Left	8
CenterLeft	9
TopLeft	10
BottomLeft	11
Right	12
CenterRight	13
TopRight	14
BottomRight	15

Transform constants:

Not supported for Cocoa. Please use DrawIconCGContext there.

See also:

- 3.3.1 DrawIcon(g as graphics,x as integer,y as integer,width as integer,height as integer) 147
- 3.3.1 DrawIcon(g as graphics,x as integer,y as integer,width as integer,height as integer,align as integer)

None	0
Disabled	1
Offline	2
Open	3
Label1	& h0100
Label2	& h0200
Label3	& h0300
Label4	& h0400
Label5	& h0500
Label6	& h0600
Label7	& h0700
Selected	& h4000
SelectedDisabled	& h4001
SelectedOffline	& h4002
SelectedOpen	& h4003

148

DrawIconCGContext(CGContextHandle as integer,x as integer,y as integer,width as integer,height as integer, align as integer, transform as integer, flags as integer, labelColor as color)

Plugin Version: 8.1 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** Draws the icon in a CoreGraphics Context.

Example:

Function GetIconImage(i as iconmbs, w as integer, h as integer) As picture
dim c as new CGPictureContextMBS(w,h)

```
const DrawNormal=0
const DrawNoImage=2
const DrawNoMask=4
const DrawSelected=& h8000
```

```
i.DrawIconCGContext(c.Handle, 0,0,w,h,0,0,DrawNoMask,& c000000)
```

```
c.Flush
```

```
Return c.CopyPicture
End Function
```

Notes:

You must make sure that the CGContext handle you pass in is valid. You can use CGContextMBS class for this and use GetCurrentCGContextMBS or Window.CGContextMBS to get a context. Please note that coordinates have the origin typically on the lower left.

Flags:

DrawNormal	0
DrawNoImage	2
DrawNoMask	4
DrawSelected	32768

Align constants:

None	0
VerticalCenter	1
Top	2
Bottom	3
HorizontalCenter	4
AbsoluteCenter	5
CenterTop	6
CenterBottom	7
Left	8
CenterLeft	9
TopLeft	10
BottomLeft	11
Right	12
CenterRight	13
TopRight	14
BottomRight	15

Transform constants:

GetBackground as IconMBS

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** If the icon is a composited one, this function returns the icon used for the background.

Notes:

Returns nil on any error.

Lasterror ist set.

None	0
Disabled	1
Offline	2
Open	3
Label1	& h0100
Label2	& h0200
Label3	& h0300
Label4	& h0400
Label5	& h0500
Label6	& h0600
Label7	& h0700
Selected	& h4000
SelectedDisabled	& h4001
SelectedOffline	& h4002
SelectedOpen	& h4003

GetForeground as IconMBS

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** If the icon is a composited one, this function returns the icon used for the foreground.

Notes:

Returns nil on any error.
LastError ist set.

IconFamily as IconFamilyMBS

Plugin Version: 2.6 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** Returns the icon converted to an Iconfamily.

Example:

```
dim i as new IconMBS("FNDR", "MACS")
```

```
Backdrop = i.IconFamily.Thumbnail32BitData
```

IsIconRefMaskEmpty as boolean

Plugin Version: 2.6 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** Returns true if the mask of the icon is empty.

Example:

```
dim i as new IconMBS("FNDR", "MACS")
```

```
MsgBox str(i.IsIconRefMaskEmpty)
```

Notes: Lasterror is set.

PointInIcon(pointx as integer,pointy as integer,x as integer,y as integer,width as integer,height as integer,align as integer) as boolean

Plugin Version: 2.6 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** Tests whether a point is inside the icon's picture.

Notes:

The coordinates for pointx/pointy and x/y must be in the same system.

Align constants:

RectInIcon(rectx as integer,recty as integer,rectwidth as integer,rectheight as integer,x as integer,y as integer,width as integer,height as integer,align as integer) as boolean

Plugin Version: 2.6 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** Tests whether a rectangle is inside the icon's picture.

Notes:

The coordinates for both rectangles must be in the same coordinate system.

Align constants:

This call may fail in some RB versions because of the count of parameters.

None	0
VerticalCenter	1
Top	2
Bottom	3
HorizontalCenter	4
AbsoluteCenter	5
CenterTop	6
CenterBottom	7
Left	8
CenterLeft	9
TopLeft	10
BottomLeft	11
Right	12
CenterRight	13
TopRight	14
BottomRight	15

RetainCount as integer

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** How many references to this icon are hold on this Mac.

Example:

```
dim i as new IconMBS("FNDR", "MACS") // Finder Icon  
  
MsgBox str(i.RetainCount)
```

3.3.2 Properties

handle as integer

Plugin Version: 2.6 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** The handle of this icon in memory.

Example:

```
dim i as new IconMBS("FNDR", "MACS") // Finder Icon
```

None	0
VerticalCenter	1
Top	2
Bottom	3
HorizontalCenter	4
AbsoluteCenter	5
CenterTop	6
CenterBottom	7
Left	8
CenterLeft	9
TopLeft	10
BottomLeft	11
Right	12
CenterRight	13
TopRight	14
BottomRight	15

```
MsgBox str(i.handle)
```

Notes: (Read and Write property)

LastError as integer

Plugin Version: 2.7 Console & Web: No Mac: Yes, Win: Yes, Linux: No, . **Function:** The last error code.

Example:

```
dim i as new IconMBS("FNDR", "MACS") // Finder Icon
```

```
MsgBox str(i.LastError)
```

Notes:

The last function was successful if lasterror is 0.

If the last function was not available on this machine, the value is set to -1.

Other values are Mac OS error codes.

(Read and Write property)

Release as boolean

Plugin Version: 2.6 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** whether the destructor will release the handle.

Example:

```
dim i as new IconMBS("FNDR", "MACS") // Finder Icon
```

```
MsgBox str(i.Release)
```

Notes: (Read and Write property)

valid as boolean

Plugin Version: 2.6 Console & Web: No Mac: Yes, Win: No, Linux: No, . **Function:** Were the constructors successful?

Example:

```
dim i as new IconMBS("FNDR", "MACS") // Finder Icon
```

```
MsgBox str(i.valid)
```

Notes: (Read and Write property)

Chapter 4

Mac

4.1 Globals

SetDesktopPictureMBS(file as folderitem) as integer

Plugin Version: 3.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: No, . **Function:** Asks the Finder/Explorer to change the desktop picture.

Notes:

File must be a valid folderitem for an existing file to define a new desktop picture.
Returns a Mac OS or Windows error code or -1 if the function is not available.

You can use file=nil to remove the desktop wallpaper on Windows.

Chapter 5

Pictures Import and Export

5.1 Globals

`BinaryStringtoPictureMBS(data as String) as Picture`

Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates the picture back from the binary data inside the string.

Example:

```
dim pic as Picture = LogoMBS(500)

// encode
dim s as string = PicturetoBinaryStringMBS(pic)

// decode
Backdrop = BinaryStringtoPictureMBS(s)
```

Notes:

The format of the binary encoded picture data:

- + 0 Kenn, PPIC for Packed Picture
- + 4 Length of whole block
- + 8 Width (BigEndian)

+12 Height (BigEndian)
 +16 Depth (BigEndian, 32 for 32bit)
 +20 Offset of the binary data. maybe 40.
 +24 Reserved for future use. Should be 0.
 +40 Pixel Data, packed R, G, B in one byte per Subpixel.

300x300 Pixels will make up $300*300+40 \rightarrow 270040$ Bytes.

This method does not require Quicktime or any other OS Service, but it does no compression.

May be a good way to store pictures crossplatform inside a database. As Valentina can do its own Zip based compression, this may be a wonderful way to store pictures uncompressed (or lossless compressed) inside the database.

BMPStringtoPictureMBS(data as string) as picture

Plugin Version: 8.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Reads a BMP picture from a string.

Example:

```
dim p as Picture = LogoMBS(500)
dim s as string = p.BMPDataMBS
dim q as Picture = BMPStringtoPictureMBS(s)
window1.Backdrop = q
```

Notes:

This function is endian safe and supports 1, 4, 8, 16, 24, 32 bit BMP images.
 For 32bit images the alpha value is ignored.
 Returns nil on any error.
 Only uncompressed BMP files are supported.

MergePictureMBS(source1 as picture, source2 as picture) as picture

Plugin Version: 7.5 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Merges the two pictures into one.

Example:

// in RB this method would work like this:

```

dim i,j as integer
dim col2 as color
dim r1,r2,g1,g2,b1,b2 as integer
dim dest as Picture // destination
dim source1, source2 as Picture // source pictures

col2 = source1.graphics.pixel(i,j)
r1 = col2.red
g1 = col2.green
b1 = col2.blue
col2 = source2.graphics.pixel(i,j)
r2 = col2.red
g2 = col2.green
b2 = col2.blue

dest.graphics.pixel(i,j) = RGB(max(r1,r2), max(g1,g2), max(b1,b2))

```

Notes:

Masks are ignored.

Returns nil on low memory.

Both pictures must have the same size and not be nil.

PicturetoBinaryStringMBS(p as picture) as string

Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a string with the picture content for saving.

Example:

```

dim s as string
dim pic as picture = LogoMBS(100)

s=PicturetoBinaryStringMBS(pic)

```

Notes:

The format of the binary encoded picture data:

- + 0 Kenn, PPIC for Packed Picture
- + 4 Length of whole block
- + 8 Width (BigEndian)
- +12 Height (BigEndian)
- +16 Depth (BigEndian, 32 for 32bit)
- +20 Offset of the binary data. maybe 40.
- +24 Reserved for future use. Should be 0.
- +40 Pixel Data, packed R, G, B in one byte per Subpixel.

300x300 Pixels will make up $300*300+40 \rightarrow 270040$ Bytes.

This method does not require Quicktime or any other OS Service, but it does no compression.
Does not handle mask or alpha channel.

May be a good way to store pictures crossplatform inside a database. As Valentina can do its own Zip based compression, this may be a wonderfull way to store pictures uncompressed (or lossless compressed) inside the database.

The returned string has the encoding set to binary (no encoding). If you want to concat the string with another you should check the encoding. If you don't handle that RB may convert the JPEG data to UTF8 (Unicode) which will destroy it.

RenderSamplesMBS(Samples as memoryblock, SampleCount as integer, Smooth as integer, Width as integer, Height as integer, outlinewidth as integer, BackColor as color=& c88B5C4, ForeColor as color=& c274C5A, OutLineColor as color=& c203F4E) as Picture

Plugin Version: 9.7 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Renders audio samples into a picture.

Notes:

Samples has one byte for each audio value and 2 bytes for each stereo sample.

SampleCount: Number of Samples. = Samples.size/2

Smooth: How smooth the samples should be made.

Width: Width of picture

Height: Height of picture

outlinewidth: The width of the outline (0=no outline)

BackColor: The back color.

ForeColor: the fore color.

OutLineColor: The color for the outline.

Chapter 6

Screenshot

6.1 Globals

`ScreenshotDisplayMBS(index as integer) as picture`

Plugin Version: 3.2 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns the Screenshot from the display with the given index.

Example:

```
Backdrop = ScreenshotDisplayMBS(0)
```

Notes:

Index starts at 0 for the main display.
Works on Linux only for first screen.

Plugin version 10.4 added support for multiple displays on Windows.

ScreenshotFromStringMBS(Width as integer, Height as integer, RowBytes as integer, data as string) as picture

Plugin Version: 8.6 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates the picture from a string returned by ScreenshotStringMBS.

Example:

```
dim p as Picture
dim s as string
```

```
dim w,h,r as integer
```

```
s=ScreenshotStringMBS(w,h,r)
```

```
p=ScreenshotFromStringMBS(w,h,r,s)
```

```
Backdrop=p
```

Notes:

Returns nil on any error.
(for example if width, height and rowwidth doesn't fit together.)

ScreenshotMBS as picture

Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns a picture of the screen content in screen resolution.

Example:

```
dim p as picture
p=screenshotMBS
```

Notes:

For a rectangle only you can use ScreenShotRectMBS.

Plugin Version 7.2 adds Windows Vista Support.

ScreenshotRectMBS(left as integer, top as integer, width as integer, height as integer) as picture

Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns a picture of the screen rectangle in screen resolution.

Example:

```
dim p as picture
p=ScreenshotRectMBS(100,100,200,200)
```

Notes:

Improved in Version 3.2 to support multiple displays on Mac OS.

Plugin Version 10.4 adds Windows support.

See also:

- 6.1 ScreenshotRectMBS(left as integer, top as integer, width as integer, height as integer, destwidth as integer, destheight as integer) as picture 167

ScreenshotRectMBS(left as integer, top as integer, width as integer, height as integer, destwidth as integer, destheight as integer) as picture

Plugin Version: 6.3 Console & Web: Yes Mac: Yes, Win: Yes, Linux: No, . **Function:** Returns a picture of the screen rectangle in screen resolution and scales it down to the requested size.

Example:

```
dim p as picture
p=ScreenshotRectMBS(100,100,200,200,50,50)
```

Notes:

Only for Mac OS.

On Windows or Linux, please use the other ScreenshotRectMBS without the extra parameters and scale the image yourself with the scale method needed.

This function is just to do the grab and scale in one rush to save CPU time.

Does not work on Mac OS X 10.7.

See also:

- 6.1 ScreenshotRectMBS(left as integer, top as integer, width as integer, height as integer) as picture 167

ScreenshotStringDisplayMBS(byref Width as integer, byref Height as integer, byref RowBytes as integer, index as integer) as string

Plugin Version: 8.6 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns a picture of the screen content in screen resolution.

Example:

```
dim s as string
dim w,h,r as integer
dim index as integer=0
```

```
s=ScreenshotStringDisplayMBS(w,h,r, index)
```

Notes:

Returns nil on any error.

Use ScreenshotFromStringMBS to get the picture from the string.

ScreenshotStringMBS(byref Width as integer, byref Height as integer, byref RowBytes as integer) as string

Plugin Version: 8.6 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Returns a picture of the screen content in screen resolution.

Example:

```
dim s as string
dim w,h,r as integer
s=ScreenshotStringMBS(w,h,r)
```

Notes:

Returns nil on any error.

Use `ScreenshotFromStringMBS` to get the picture from the string.

Chapter 7

X-Face

7.1 Globals

PictureFromXFaceMemoryBlockMBS(xface as memoryblock) as picture

Plugin Version: 3.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new picture from a X-Face string inside a memoryblock.

Notes:

Returns nil on any error.

The returned picture is 32 bit depth.

See also:

- 7.1 PictureFromXFaceMemoryBlockMBS(xface as memoryblock, size as integer) as picture 171

PictureFromXFaceMemoryBlockMBS(xface as memoryblock, size as integer) as picture

Plugin Version: 3.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new picture from a X-Face string inside a memoryblock with the given size.

Notes:

Returns nil on any error.

The returned picture is 32 bit depth.

See also:

- 7.1 PictureFromXFaceMemoryBlockMBS(xface as memoryblock) as picture 171

PictureFromXFaceStringMBS(xface as string) as picture

Plugin Version: 3.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a new picture from a X-Face string.

Notes:

Returns nil on any error.
The returned picture is 32 bit depth.

XFaceStringFromPictureMBS(pic as picture) as string

Plugin Version: 3.1 Console & Web: Yes Mac: Yes, Win: Yes, Linux: Yes, . **Function:** Creates a X-Face encoded string with the picture as content.

Notes:

The picture is first converted to a 32 bit picture. Than it's encoded. A Pixel like c=rgb(r,g,b) is white if $(r+g+b) \geq 3*128$.

Returns "" on any error.

Chapter 8

List of all classes

• BarcodeScannerMBS	116
• IconFamilyMBS	128
• IconMBS	143
• PaletteCalculatorMBS	121
• PaletteMBS	120
• PictureConvolutionMBS	76
• PictureEditorMBS	81
• PictureLut3DMBS	101
• PictureMatrix3DMBS	104
• PictureMatrixMBS	98
• PictureMinMaxMBS	107
• PictureReaderMBS	84
• PictureSepiaMBS	94
• PictureWriterMBS	89

Chapter 9

List of all global methods

- 5.1 BinaryStringtoPictureMBS(data as String) as Picture 159
- 2.1 BlendPicturesMBS(result as picture, source as picture, sourcepercent as double, dest as picture, destpercent as double, x As Integer, y As Integer, width As Integer, height As Integer) as boolean 15
- 2.1 BlendPicturesMBS(source as picture, sourcepercent as double, dest as picture, destpercent as double) as picture 16
- 2.1 BlendPicturesWithMaskMBS(result as picture, source as picture, dest as picture, mask as picture, x As Integer, y As Integer, width As Integer, height As Integer) as boolean 16
- 2.1 BlendPicturesWithMaskMBS(source as picture, dest as picture, mask as picture) as picture 17
- 2.1 BlendPicturesWithMaskWithBackgroundMBS(SourceImage As Picture, DestImage As Picture, Mask As Picture, Result As Picture, X As Integer, Y As Integer, Width As Integer, Height As Integer) as boolean 18
- 2.1 BlendPicturesWithMaskWithBackgroundMBS(SourceImage As Picture, DestImage As Picture, Mask As Picture, Result As Picture, X As Integer, Y As Integer, Width As Integer, Height As Integer, BackgroundColour As Color) as boolean 18
- 5.1 BMPStringtoPictureMBS(data as string) as picture 160
- 2.1 ColorizePictureMBS(Pict As Picture, Mask As Picture, foreR as double, foreG as double, foreB as double, foreA as double, backR as double, backG as double, backB as double, backA as double) as boolean 19
- 2.1 CombinePicturesMBS(red as picture, blue as picture, green as picture) as picture 19
- 3.1 CompositeIconsMBS(ForeGround as IconMBS, BackGround as IconMBS) as IconMBS 127
- 2.1 DiffPicturesMBS(source as picture, dest as picture, square as boolean) as picture 19
- 2.1 GetMBfromPictureMBS(pic as picture, mask as picture, mode as string) as memoryblock 20

- 2.1 GetMBfromPictureMBS(pic as picture, mode as string) as memoryblock 21
- 2.1 MandelbrotSetMBS(Threaded as integer, width as integer, height as integer, fx as double = 4.0, fy as double = 4.0, dx as double = -2.0, dy as double = -2.0) as picture 21
- 2.1 MemoryblockABGRtoPictureMBS(dest as picture, source as memoryblock, offset as integer, width as integer, height as integer) as picture 22
- 2.1 MemoryblockABGRtoPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer) as picture 22
- 2.1 MemoryblockARGBtoPictureMBS(dest as picture, source as memoryblock, offset as integer, width as integer, height as integer) as picture 23
- 2.1 MemoryblockARGBtoPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer) as picture 24
- 2.1 MemoryblockARGBtoPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer, LittleEndian as boolean) as picture 24
- 2.1 MemoryblockBGRtoPictureMBS(dest as picture, source as memoryblock, offset as integer, width as integer, height as integer) as picture 26
- 2.1 MemoryblockBGRtoPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer) as picture 26
- 2.1 MemoryblockBGRtoPictureMBS(dest as picture, source as memoryblock, offset as integer, width as integer, height as integer) as picture 27
- 2.1 MemoryblockBGRtoPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer) as picture 28
- 2.1 MemoryblockGrayToPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer, PixelByteSize as integer) as picture 29
- 2.1 MemoryblockGrayToPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer, PixelByteSize as integer, Red as integer, Blue as integer, Green as integer) as picture 30
- 2.1 MemoryblockGrayToPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer, PixelByteSize as integer, Red() as integer, Blue() as integer, Green() as integer) as picture 31
- 2.1 MemoryblockRGBtoPictureMBS(dest as picture, source as memoryblock, offset as integer, width as integer, height as integer, FlipVertically as boolean=false) as picture 31
- 2.1 MemoryblockRGBtoPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer, FlipVertically as boolean=false) as picture 32
- 2.1 MemoryblockRGBtoPictureMBS(dest as picture, source as memoryblock, offset as integer, width as integer, height as integer) as picture 33
- 2.1 MemoryblockRGBtoPictureMBS(source as memoryblock, offset as integer, width as integer, height as integer) as picture 34
- 5.1 MergePictureMBS(source1 as picture, source2 as picture) as picture 160

- 2.1 NewBluePaletteMBS as PaletteMBS 35
- 2.1 NewGrayPaletteMBS as PaletteMBS 36
- 2.1 NewGreenPaletteMBS as PaletteMBS 36
- 3.1 NewIconFamilyMBS as IconFamilyMBS 128
- 3.1 NewIconFamilyMBSFromScrap as IconFamilyMBS 128
- 2.1 NewPaletteMBS(count as integer) as PaletteMBS 36
- 2.1 NewPalmPaletteMBS as PaletteMBS 37
- 2.1 NewPictureEditorMBS(pic as picture) as PictureEditorMBS 37
- 2.1 NewPictureMBS(width as integer, height as integer, pixeltype as integer, buffer as memoryblock, rowbytes as integer) as picture 38
- 2.1 NewPictureReaderMBS(pic as picture) as PictureReaderMBS 38
- 2.1 NewPictureWithColorMBS(width as integer, height as integer, c as color) as picture 40
- 2.1 NewPictureWriterMBS(width as integer, height as integer) as PictureWriterMBS 40
- 2.1 NewRedPaletteMBS as PaletteMBS 41
- 2.1 NewSystemPaletteMBS as PaletteMBS 42
- 2.1 NewWebPaletteMBS as PaletteMBS 42
- 2.1 NewWindowsPaletteMBS as PaletteMBS 43
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean) as boolean 43
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color) as boolean 45
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color, MaskColour As color) as boolean 48
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer) as boolean 49
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer, MaskColour As Integer) as boolean 51
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean) as boolean 54

- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color) as boolean 56
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As color, MaskColour As color) as boolean 58
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer) as boolean 60
- 2.1 PictureCombineMBS(DestImage As Picture, Image As Picture, PreMultipliedSource as boolean, Mask As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer, UseColours As Boolean, ForeColour As Integer, MaskColour As Integer) as boolean 62
- 2.1 PictureCopyPixelFastMBS(DestImage As Picture, Source As Picture, DestX As Integer, DestY As Integer, SourceX As Integer, SourceY As Integer, Width As Integer, Height As Integer) as boolean 63
- 7.1 PictureFromXFaceMemoryBlockMBS(xface as memoryblock) as picture 171
- 7.1 PictureFromXFaceMemoryBlockMBS(xface as memoryblock, size as integer) as picture 171
- 7.1 PictureFromXFaceStringMBS(xface as string) as picture 172
- 5.1 PicturetoBinaryStringMBS(p as picture) as string 161
- 2.1 PtrABGRtoPictureMBS(dest as picture, source as Ptr, offset as integer, width as integer, height as integer) as picture 65
- 2.1 PtrABGRtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer) as picture 65
- 2.1 PtrARGBtoPictureMBS(dest as picture, source as Ptr, offset as integer, width as integer, height as integer) as picture 66
- 2.1 PtrARGBtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer) as picture 66
- 2.1 PtrARGBtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer, LittleEndian as boolean) as picture 67
- 2.1 PtrBGRtoPictureMBS(dest as picture, source as Ptr, offset as integer, width as integer, height as integer) as picture 68
- 2.1 PtrBGRtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer) as picture 68
- 2.1 PtrBGRtoPictureMBS(dest as picture, source as Ptr, offset as integer, width as integer, height as integer) as picture 69

- 2.1 PtrBGRtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer) as picture 69
- 2.1 PtrGrayToPictureMBS(source as Ptr, offset as integer, width as integer, height as integer, Pixel-ByteSize as integer) as picture 70
- 2.1 PtrGrayToPictureMBS(source as Ptr, offset as integer, width as integer, height as integer, Pixel-ByteSize as integer, Red as integer, Blue as integer, Green as integer) as picture 70
- 2.1 PtrGrayToPictureMBS(source as Ptr, offset as integer, width as integer, height as integer, Pixel-ByteSize as integer, Red() as integer, Blue() as integer, Green() as integer) as picture 71
- 2.1 PtrRGBAtoPictureMBS(dest as picture, source as Ptr, offset as integer, width as integer, height as integer, FlipVertically as boolean=false) as picture 72
- 2.1 PtrRGBAtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer, FlipVertically as boolean=false) as picture 73
- 2.1 PtrRGBtoPictureMBS(dest as picture, source as Ptr, offset as integer, width as integer, height as integer) as picture 73
- 2.1 PtrRGBtoPictureMBS(source as Ptr, offset as integer, width as integer, height as integer) as picture 74
- 5.1 RenderSamplesMBS(Samples as memoryblock, SampleCount as integer, Smooth as integer, Width as integer, Height as integer, outlinewidth as integer, BackColor as color=& c88B5C4, ForeColor as color=& c274C5A, OutLineColor as color=& c203F4E) as Picture 162
- 6.1 ScreenshotDisplayMBS(index as integer) as picture 165
- 6.1 ScreenshotFromStringMBS(Width as integer, Height as integer, RowBytes as integer, data as string) as picture 166
- 6.1 ScreenshotMBS as picture 166
- 6.1 ScreenshotRectMBS(left as integer, top as integer, width as integer, height as integer) as picture 167
- 6.1 ScreenshotRectMBS(left as integer, top as integer, width as integer, height as integer, destwidth as integer, destheight as integer) as picture 167
- 6.1 ScreenshotStringDisplayMBS(byref Width as integer, byref Height as integer, byref RowBytes as integer, index as integer) as string 168
- 6.1 ScreenshotStringMBS(byref Width as integer, byref Height as integer, byref RowBytes as integer) as string 168
- 4.1 SetDesktopPictureMBS(file as folderitem) as integer 157
- 2.1 TintPictureMBS(source as picture, GreyBase as color, SepiaBase as color) as picture 74
- 7.1 XFaceStringFromPictureMBS(pic as picture) as string 172