PAlib

Generated by Doxygen 1.6.1

Thu Dec 17 18:40:19 2009

Contents

1 PAlib 0911XX Documentation **1**

1.1 Introduction . . . 1
1.2 Core library . . . 1
1.3 Tiled backgrounds . . . 1
1.4 Bitmapped backgrounds . . . 1
1.5 Sprites . . . 2
1.6 Palettes . . . 2
1.7 Input . . . 2
1.8 Sound . . . 2
1.9 Misc. . . . 2

2 Deprecated List **3**

3 Module Documentation **7**

3.1 16color pseudo-bitmap mode . . . 7
3.1.1 Detailed Description . . . 8
3.1.2 Define Documentation . . . 8
3.1.2.1 PA_16cCustomFont . . . 8
3.1.3 Function Documentation . . . 9
3.1.3.1 PA_Init16cBg . . . 9
3.1.3.2 PA_16cErase . . . 9
3.1.3.3 PA_InitComplete16c . . . 9
3.1.3.4 PA_16cText . . . 9
3.1.3.5 PA_Add16cFont . . . 10
3.1.3.6 PA_16cPutPixel . . . 10
3.1.3.7 PA_16c8X4 . . . 10

3.1.3.8 PA_16c8X6 . . . 11
3.1.3.9 PA_16c8X8 . . . 11
3.1.3.10 PA_16c8Xi . . . 11
3.1.3.11 PA_16cClearZone . . . 12
3.1.3.12 PA_16cGetPixel . . . 12
3.2 3D Sprite System . . . 13
3.2.1 Detailed Description . . . 16
3.2.2 Function Documentation . . . 16
3.2.2.1 PA_3DStartSpriteAnimEx . . . 16
3.2.2.2 PA_3DStartSpriteAnim . . . 17
3.2.2.3 PA_3DStopSpriteAnim . . . 17
3.2.2.4 PA_3DSetSpriteAnimFrame . . . 17
3.2.2.5 PA_3DGetSpriteAnimFrame . . . 17
3.2.2.6 PA_3DSetSpriteAnimSpeed . . . 18
3.2.2.7 PA_3DGetSpriteAnimSpeed . . . 18
3.2.2.8 PA_3DSetSpriteNCycles . . . 18
3.2.2.9 PA_3DGetSpriteNCycles . . . 18
3.2.2.10 PA_3DSpriteAnimPause . . . 18
3.3 Old large background system . . . 19
3.3.1 Detailed Description . . . 20
3.3.2 Define Documentation . . . 20
3.3.2.1 PA_LoadLargeBg . . . 20
3.3.2.2 PA_LoadPAGfxLargeBg . . . 20
3.3.2.3 PA_LoadLargeBgEx . . . 21
3.3.3 Function Documentation . . . 21
3.3.3.1 PA_InfLargeScrollX . . . 21
3.3.3.2 PA_InfLargeScrollY . . . 22
3.3.3.3 PA_InfLargeScrollXY . . . 22
3.3.3.4 PA_LargeScrollX . . . 22
3.3.3.5 PA_LargeScrollY . . . 23
3.3.3.6 PA_LargeScrollXY . . . 23
3.4 Rotating Backgrounds . . . 24
3.4.1 Detailed Description . . . 24
3.4.2 Define Documentation . . . 24

3.4.2.1 PA_LoadRotBg 24
3.4.2.2 PA_LoadPAGfxRotBg 25
3.4.3 Function Documentation 25
3.4.3.1 PA_SetBgRot 25
3.5 Tiled Background Modes 27
3.5.1 Detailed Description 30
3.5.2 Define Documentation 30
3.5.2.1 PA_HideBg 30
3.5.2.2 PA_ShowBg 31
3.5.2.3 PA_ResetBg 31
3.5.2.4 PA_LoadBgTiles 31
3.5.2.5 PA_LoadTiledBg 31
3.5.2.6 PA_LoadSimpleBg 32
3.5.2.7 PA_LoadBg 32
3.5.2.8 PA_SetMapTileAll 33
3.5.2.9 PA_EasyBgLoad 34
3.5.2.10 PA_EasyBgLoadPtr 34
3.5.3 Enumeration Type Documentation 34
3.5.3.1 "@0 34
3.5.4 Function Documentation 35
3.5.4.1 PA_ResetBgSysScreen 35
3.5.4.2 PA_InitBg 35
3.5.4.3 PA_LoadBgTilesEx 35
3.5.4.4 PA_ReLoadBgTiles 36
3.5.4.5 PA_DeleteTiles 36
3.5.4.6 PA_DeleteMap 36
3.5.4.7 PA_DeleteBg 36
3.5.4.8 PA_LoadBgMap 37
3.5.4.9 PA_LoadBackground 37
3.5.4.10 PA_BGScrollX 37
3.5.4.11 PA_BGScrollY 37
3.5.4.12 PA_SetMapTile 38
3.5.4.13 PA_SetLargeMapTile 38
3.5.4.14 PA_SetMapTileHflip 38

3.5.4.15 PA_SetMapTileVflip . . . 39
3.5.4.16 PA_SetMapTilePal . . . 39
3.5.4.17 PA_SetBgPrio . . . 39
3.5.4.18 PA_SetBgPrioSeq . . . 39
3.5.4.19 PA_ClearBg . . . 40
3.5.4.20 PA_EasyBgScrollX . . . 40
3.5.4.21 PA_EasyBgScrollY . . . 40
3.5.4.22 PA_EasyBgScrollXY . . . 40
3.5.4.23 PA_EasyBgGetPixel . . . 41
3.5.4.24 PA_EasyBgGetPixelCol . . . 41
3.5.4.25 PA_SetBgWrap . . . 41
3.5.4.26 PA_InitParallaxX . . . 41
3.5.4.27 PA_InitParallaxY . . . 42
3.5.4.28 PA_ParallaxScrollX . . . 42
3.5.4.29 PA_ParallaxScrollY . . . 42
3.5.4.30 PA_ParallaxScrollXY . . . 43
3.6 Background Transition Effects . . . 44
3.6.1 Detailed Description . . . 44
3.6.2 Function Documentation . . . 44
3.6.2.1 PA_InitBgTransEx . . . 44
3.6.2.2 PA_InitBgTrans . . . 44
3.6.2.3 PA_BgTransUpDown . . . 45
3.6.2.4 PA_BgTransLeftRight . . . 45
3.6.2.5 PA_BgTransDiag . . . 45
3.6.2.6 PA_BgTransCenter . . . 45
3.7 Debugging utilities . . . 47
3.7.1 Detailed Description . . . 47
3.7.2 Define Documentation . . . 47
3.7.2.1 PA_Assert . . . 47
3.7.3 Function Documentation . . . 47
3.7.3.1 PA_iDeaS_DebugOutput . . . 47
3.7.3.2 PA_iDeaS_DebugPrintf . . . 48
3.8 Bitmap mode . . . 49
3.8.1 Detailed Description . . . 51

3.8.2 Define Documentation 51
3.8.2.1 PA_Get16bitPixel 51
3.8.2.2 PA_SetDrawSize 51
3.8.2.3 PA_Load8bitBitmap 52
3.8.2.4 PA_Load16bitBitmap 52
3.8.2.5 PA_Clear8bitBg 52
3.8.2.6 PA_Clear16bitBg 52
3.8.3 Function Documentation 52
3.8.3.1 PA_Init8bitBg 52
3.8.3.2 PA_InitBig8bitBg 53
3.8.3.3 PA_Init16bitBg 53
3.8.3.4 PA_Put8bitPixel 53
3.8.3.5 PA_Put2_8bitPixels 53
3.8.3.6 PA_PutDouble8bitPixels 54
3.8.3.7 PA_Put4_8bitPixels 54
3.8.3.8 PA_Get8bitPixel 54
3.8.3.9 PA_Put16bitPixel 55
3.8.3.10 PA_Draw8bitLine 55
3.8.3.11 PA_Draw16bitLine 55
3.8.3.12 PA_Draw16bitLineEx 56
3.8.3.13 PA_Draw8bitLineEx 56
3.8.3.14 PA_Draw16bitRect 56
3.8.3.15 PA_8bitDraw 57
3.8.3.16 PA_16bitDraw 57
3.8.3.17 PA_LoadJpeg 57
3.8.3.18 PA_LoadBmpToBuffer 58
3.8.3.19 PA_LoadBmpEx 58
3.8.3.20 PA_LoadBmp 58
3.8.3.21 PA_GetBmpWidth 58
3.8.3.22 PA_GetBmpHeight 59
3.9 Fake 16bit bitmap mode 60
3.9.1 Detailed Description 61
3.9.2 Define Documentation 61
3.9.2.1 PA_LoadFake16bitBitmap 61

3.9.2.2 PA_ClearFake16bitBg 61
3.9.2.3 PA_PutFake16bitPixel 61
3.9.2.4 PA_GetFake16bitPixel 62
3.9.2.5 PA_DrawFake16bitRect 62
3.9.2.6 PA_Fake16bitLoadBmpEx 62
3.9.2.7 PA_Fake16bitLoadBmp 63
3.9.2.8 PA_Fake16bitLoadGif 63
3.9.2.9 PA_Fake16bitLoadJpeg 63
3.9.3 Function Documentation 63
3.9.3.1 PA_InitFake16bitBg 63
3.9.3.2 PA_DrawFake16bitLine 63
3.10 General Functions 65
3.10.1 Detailed Description 67
3.10.2 Define Documentation 67
3.10.2.1 PA_LegacyIPCInit 67
3.10.2.2 PA_CloseLidSound 68
3.10.2.3 PA_CloseLidSound2 68
3.10.2.4 PA_WaitFor 68
3.10.3 Enumeration Type Documentation 69
3.10.3.1 "@5 69
3.10.3.2 "@6 69
3.10.4 Function Documentation 69
3.10.4.1 PA_SetVideoMode 69
3.10.4.2 PA_SetAutoCheckLid 69
3.10.4.3 PA_SetLedBlink 70
3.10.4.4 PA_SetScreenLight 70
3.10.4.5 PA_SetDSLBrightness 70
3.10.4.6 PA_Locate 70
3.11 Gif functions 71
3.11.1 Detailed Description 71
3.11.2 Function Documentation 71
3.11.2.1 PA_GetGifWidth 71
3.11.2.2 PA_GetGifHeight 72
3.11.2.3 PA_LoadGifXY 72

3.11.2.4 PA_LoadGif 72
3.11.2.5 PA_GifAnimSpeed 72
3.11.2.6 PA_GifAnimStop 72
3.11.2.7 PA_GifSetStartFrame 73
3.11.2.8 PA_GifSetEndFrame 73
3.12 Keyboard 74
3.12.1 Detailed Description 75
3.12.2 Define Documentation 75
3.12.2.1 PA_InitCustomKeyboard 75
3.12.3 Function Documentation 75
3.12.3.1 PA_LoadDefaultKeyboard 75
3.12.3.2 PA_LoadKeyboard 76
3.12.3.3 PA_ScrollKeyboardX 76
3.12.3.4 PA_ScrollKeyboardY 76
3.12.3.5 PA_ScrollKeyboardXY 76
3.12.3.6 PA_KeyboardIn 76
3.12.3.7 PA_SetKeyboardColor 77
3.12.3.8 PA_SetKeyboardScreen 77
3.13 Key input system 78
3.13.1 Detailed Description 79
3.13.2 Define Documentation 79
3.13.2.1 PA_MoveSprite 79
3.13.2.2 PA_StylusInZone 79
3.13.3 Function Documentation 80
3.13.3.1 PA_MoveSpritePix 80
3.13.3.2 PA_MoveSpriteEx 80
3.13.3.3 PA_MoveSpriteDistance 80
3.13.3.4 PA_SpriteStylusOverEx 80
3.13.3.5 PA_SpriteTouchedEx 81
3.13.3.6 PA_SpriteTouched 81
3.13.3.7 PA_SpriteStylusOver 81
3.14 Special controllers 82
3.14.1 Detailed Description 82
3.15 Math functions 83

3.15.1 Detailed Description 84
3.15.2 Function Documentation 84
3.15.2.1 PA_SRand 84
3.15.2.2 PA_RandMax 84
3.15.2.3 PA_RandMinMax 84
3.15.2.4 PA_Distance 85
3.15.2.5 PA_TrueDistance 85
3.15.2.6 PA_AdjustAngle 85
3.15.2.7 PA_GetAngle 86
3.15.2.8 PA_mulf32 86
3.15.2.9 PA_divf32 86
3.15.2.10 PA_modf32 86
3.15.2.11 PA_sqrtf32 86
3.16 Microphone 87
3.16.1 Detailed Description 87
3.16.2 Function Documentation 87
3.16.2.1 PA_MicStartRecording 87
3.16.2.2 PA_MicReplay 87
3.17 Mode 7 commands 88
3.17.1 Detailed Description 88
3.17.2 Function Documentation 88
3.17.2.1 PA_InitMode7 88
3.17.2.2 PA_Mode7Angle 89
3.17.2.3 PA_Mode7MoveLeftRight 89
3.17.2.4 PA_Mode7MoveForwardBack 89
3.17.2.5 PA_Mode7X 89
3.17.2.6 PA_Mode7Z 90
3.17.2.7 PA_Mode7SetPointXZ 90
3.17.2.8 PA_Mode7Height 90
3.18 DS Motion functions 91
3.18.1 Detailed Description 91
3.19 Palette system 92
3.19.1 Detailed Description 93
3.19.2 Define Documentation 93

3.19.2.1 PA_LoadPal 93
3.19.2.2 PA_LoadPal16 94
3.19.2.3 PA_LoadSprite16cPal 94
3.19.2.4 PA_RGB 94
3.19.2.5 PA_SetBgPalCol 95
3.19.3 Function Documentation 95
3.19.3.1 PA_Load8bitBgPal 95
3.19.3.2 PA_SetBrightness 95
3.19.3.3 PA_SetPalNeg 95
3.19.3.4 PA_SetPal16Neg 95
3.19.3.5 PA_LoadSpritePal 96
3.19.3.6 PA_LoadBgPalN 96
3.19.3.7 PA_LoadBgPal 96
3.19.3.8 PA_SetBgPalNCol 97
3.19.3.9 PA_SetBgColor 97
3.19.3.10 PA_SetSpritePalCol 97
3.19.3.11 PA_3DSetSpritePalCol 97
3.20 Palette system for Dual Screen 98
3.20.1 Detailed Description 98
3.20.2 Define Documentation 98
3.20.2.1 PA_DualLoadPal 98
3.20.2.2 PA_DualLoadPal16 99
3.20.3 Function Documentation 99
3.20.3.1 PA_DualSetPalNeg 99
3.20.3.2 PA_DualSetPal16Neg 99
3.20.3.3 PA_DualLoadSpritePal 100
3.20.3.4 PA_DualLoadBgPal 100
3.20.3.5 PA_DualSetBgColor 100
3.21 Shape Recognition 101
3.21.1 Detailed Description 101
3.21.2 Function Documentation 101
3.21.2.1 PA_RecoAddShape 101
3.21.2.2 PA_UsePAGraffiti 101
3.22 Special Effects 102

3.22.1 Detailed Description . . . 102
3.22.2 Define Documentation . . . 102
3.22.2.1 PA_EnableBgMosaic . . . 102
3.22.2.2 PA_DisableBgMosaic . . . 103
3.22.2.3 PA_SetBgMosaicXY . . . 103
3.22.2.4 PA_SetSpriteMosaicXY . . . 103
3.22.2.5 PA_EnableSpecialFx . . . 103
3.22.2.6 PA_DisableSpecialFx . . . 104
3.22.2.7 PA_SetSFXAlpha . . . 104
3.23 Sprite system . . . 105
3.23.1 Detailed Description . . . 110
3.23.2 Define Documentation . . . 111
3.23.2.1 PA_UpdateSpriteGfx . . . 111
3.23.2.2 PA_SetSpriteRotEnable . . . 111
3.23.2.3 PA_SetSpriteRotDisable . . . 111
3.23.2.4 PA_SetSpriteX . . . 111
3.23.2.5 PA_GetSpriteX . . . 112
3.23.2.6 PA_SetSpriteY . . . 112
3.23.2.7 PA_GetSpriteY . . . 112
3.23.2.8 PA_SetSpritePal . . . 112
3.23.2.9 PA_GetSpritePal . . . 113
3.23.2.10 PA_SetSpriteDblsize . . . 113
3.23.2.11 PA_GetSpriteDblsize . . . 113
3.23.2.12 PA_SetSpriteColors . . . 113
3.23.2.13 PA_GetSpriteColors . . . 114
3.23.2.14 PA_SetSpriteMode . . . 114
3.23.2.15 PA_GetSpriteMode . . . 114
3.23.2.16 PA_SetSpriteMosaic . . . 114
3.23.2.17 PA_GetSpriteMosaic . . . 115
3.23.2.18 PA_SetSpriteHflip . . . 115
3.23.2.19 PA_GetSpriteHflip . . . 115
3.23.2.20 PA_SetSpriteVflip . . . 115
3.23.2.21 PA_GetSpriteVflip . . . 116
3.23.2.22 PA_SetSpriteGfx . . . 116

3.23.2.23 PA_GetSpriteGfx 116
3.23.2.24 PA_SetSpritePrio 116
3.23.2.25 PA_GetSpritePrio 117
3.23.2.26 PA_GetSpriteLx 117
3.23.2.27 PA_GetSpriteLy 117
3.23.2.28 PA_CloneSprite 117
3.23.3 Function Documentation 118
3.23.3.1 PA_CreateGfx 118
3.23.3.2 PA_CreateSprite 118
3.23.3.3 PA_CreateSpriteEx 118
3.23.3.4 PA_Create16bitSpriteEx 119
3.23.3.5 PA_Create16bitSpriteFromGfx 120
3.23.3.6 PA_Create16bitSprite 120
3.23.3.7 PA_CreateSpriteFromGfx 121
3.23.3.8 PA_CreateSpriteExFromGfx 121
3.23.3.9 PA_UpdateGfx 122
3.23.3.10 PA_UpdateGfxAndMem 122
3.23.3.11 PA_DeleteGfx 122
3.23.3.12 PA_DeleteSprite 122
3.23.3.13 PA_SetRotset 123
3.23.3.14 PA_SetRotsetNoZoom 123
3.23.3.15 PA_SetRotsetNoAngle 123
3.23.3.16 PA_SetSpriteXY 124
3.23.3.17 PA_Set16bitSpriteAlpha 124
3.23.3.18 PA_SetSpriteAnimEx 124
3.23.3.19 PA_SetSpriteAnim 124
3.23.3.20 PA_StartSpriteAnimEx 125
3.23.3.21 PA_StartSpriteAnim 125
3.23.3.22 PA_StopSpriteAnim 126
3.23.3.23 PA_SetSpriteAnimFrame 126
3.23.3.24 PA_GetSpriteAnimFrame 126
3.23.3.25 PA_SetSpriteAnimSpeed 126
3.23.3.26 PA_GetSpriteAnimSpeed 127
3.23.3.27 PA_SetSpriteNCycles 127

3.23.3.28 PA_GetSpriteNCycles 127
3.23.3.29 PA_SpriteAnimPause 127
3.23.3.30 PA_SetSpritePixel 128
3.23.3.31 PA_GetSpritePixel 128
3.23.3.32 PA_GetSprite16cPixel 128
3.23.3.33 PA_InitSpriteDraw 128
3.23.3.34 PA_InitSpriteExtPrio 129
3.24 Sprite system for Dual Screen 130
3.24.1 Detailed Description 133
3.24.2 Function Documentation 133
3.24.2.1 PA_SetScreenSpace 133
3.24.2.2 PA_DualSetSpriteX 133
3.24.2.3 PA_DualSetSpriteY 133
3.24.2.4 PA_DualSetSpriteXY 133
3.24.2.5 PA_DualCreateSprite 134
3.24.2.6 PA_DualCreateSpriteEx 134
3.24.2.7 PA_DualCreate16bitSpriteEx 135
3.24.2.8 PA_DualCreate16bitSprite 135
3.24.2.9 PA_DualCreateSpriteFromGfx 136
3.24.2.10 PA_DualCreateSpriteExFromGfx 136
3.24.2.11 PA_DualUpdateSpriteGfx 137
3.24.2.12 PA_DualUpdateGfx 137
3.24.2.13 PA_DualDeleteSprite 137
3.24.2.14 PA_DualSetSpriteRotEnable 137
3.24.2.15 PA_DualSetSpriteRotDisable 138
3.24.2.16 PA_DualSetRotset 138
3.24.2.17 PA_DualSetRotsetNoZoom 138
3.24.2.18 PA_DualSetRotsetNoAngle 138
3.24.2.19 PA_DualSetSpritePal 139
3.24.2.20 PA_DualSetSpriteDblsize 139
3.24.2.21 PA_DualSetSpriteColors 139
3.24.2.22 PA_DualSetSpriteMode 139
3.24.2.23 PA_DualSetSpriteMosaic 140
3.24.2.24 PA_DualSetSpriteHflip 140

3.24.2.25 PA_DualSetSpriteVflip 140
3.24.2.26 PA_DualSetSpriteGfx 140
3.24.2.27 PA_DualSetSpritePrio 141
3.24.2.28 PA_DualCloneSprite 141
3.24.2.29 PA_DualSetSpriteAnimEx 141
3.24.2.30 PA_DualSetSpriteAnim 141
3.24.2.31 PA_DualStartSpriteAnimEx 142
3.24.2.32 PA_DualStartSpriteAnim 142
3.24.2.33 PA_DualStopSpriteAnim 142
3.24.2.34 PA_DualSetSpriteAnimFrame 143
3.24.2.35 PA_DualGetSpriteAnimFrame 143
3.24.2.36 PA_DualSetSpriteAnimSpeed 143
3.24.2.37 PA_DualGetSpriteAnimSpeed 143
3.24.2.38 PA_DualSpriteAnimPause 143
3.25 Text output system 144
3.25.1 Detailed Description 146
3.25.2 Define Documentation 146
3.25.2.1 PA_SetTileLetter 146
3.25.2.2 PA_InitCustomText 146
3.25.2.3 PA_ShowFont 146
3.25.2.4 PA_8bitCustomFont 147
3.25.3 Function Documentation 147
3.25.3.1 PA_LoadDefaultText 147
3.25.3.2 PA_SetTextTileCol 147
3.25.3.3 PA_OutputText 148
3.25.3.4 PA_OutputSimpleText 148
3.25.3.5 PA_BoxText 148
3.25.3.6 PA_BoxTextNoWrap 149
3.25.3.7 PA_SetTextCol 149
3.25.3.8 PA_LoadText 149
3.25.3.9 PA_8bitText 150
3.25.3.10 PA_CenterSmartText 150
3.25.3.11 PA_AddBitmapFont 151
3.25.3.12 PA_InitTextBorders 151

3.25.3.13 PA_EraseTextBox 151
3.25.3.14 PA_SimpleBoxText 151
3.25.3.15 PA_ClearTextBg 152
3.25.3.16 PA_Print 152
3.25.3.17 PA_PrintLetter 152
3.26 Bg Modes on 2 Screens 153
3.26.1 Detailed Description 155
3.26.2 Define Documentation 155
3.26.2.1 PA_DualLoadTiledBg 155
3.26.2.2 PA_DualLoadSimpleBg 156
3.26.2.3 PA_DualLoadRotBg 157
3.26.2.4 PA_DualLoadBg 157
3.26.2.5 PA_DualLoadPAGfxLargeBg 158
3.26.2.6 PA_DualLoadLargeBg 158
3.26.2.7 PA_DualLoadLargeBgEx 159
3.26.2.8 PA_DualEasyBgLoad 160
3.26.3 Function Documentation 160
3.26.3.1 PA_DualHideBg 160
3.26.3.2 PA_DualShowBg 160
3.26.3.3 PA_DualDeleteBg 160
3.26.3.4 PA_DualBGScrollX 161
3.26.3.5 PA_DualBGScrollY 161
3.26.3.6 PA_DualBGScrollXY 161
3.26.3.7 PA_DualEasyBgScrollX 161
3.26.3.8 PA_DualEasyBgScrollY 161
3.26.3.9 PA_DualLoadBackground 162
3.26.3.10 PA_DualEasyBgScrollXY 162
3.26.3.11 PA_DualInfLargeScrollX 162
3.26.3.12 PA_DualInfLargeScrollY 162
3.26.3.13 PA_DualInfLargeScrollXY 163
3.26.3.14 PA_DualLargeScrollX 163
3.26.3.15 PA_DualLargeScrollY 163
3.26.3.16 PA_DualLargeScrollXY 163
3.26.3.17 PA_DualInitParallaxX 164

3.26.3.18 PA_DualInitParallaxY 164
3.26.3.19 PA_DualParallaxScrollX 164
3.26.3.20 PA_DualParallaxScrollY 164
3.26.3.21 PA_DualParallaxScrollXY 165
3.26.3.22 PA_DualSetBgPrio 165
3.27 Window system 166
3.27.1 Detailed Description 167
3.27.2 Define Documentation 167
3.27.2.1 PA_SetWin1XY 167
3.27.2.2 PA_EnableWin0 167
3.27.2.3 PA_DisableWin0 167
3.27.2.4 PA_EnableWin1 168
3.27.2.5 PA_DisableWin1 168
3.27.2.6 PA_DisableWinObj 168
3.27.2.7 PA_SetOutWin 168
3.27.3 Function Documentation 169
3.27.3.1 PA_EnableWinObj 169
3.27.3.2 PA_WindowFade 169
3.28 C++ wrappers 170
3.28.1 Detailed Description 170
3.29 ASlib functions 171
3.29.1 Detailed Description 173
3.29.2 Enumeration Type Documentation 173
3.29.2.1 MP3Command 173
3.29.2.2 SoundCommand 174
3.29.2.3 MP3Status 174
3.29.2.4 AS_MODE 174
3.29.2.5 AS_DELAY 175
3.29.2.6 AS_SOUNDFORMAT 175

4 Namespace Documentation **177**

4.1 PA Namespace Reference 177
4.1.1 Detailed Description 177

5 Data Structure Documentation **179**

5.1 PA::Application Class Reference 179
5.1.1 Detailed Description 179
5.2 PA::Fixed Class Reference 180
5.2.1 Detailed Description 183
5.3 PA::HandleProvider< NHANDLES > Class Template Reference . . . 184
5.3.1 Detailed Description 184
5.4 PA_BgStruct Struct Reference 185
5.4.1 Detailed Description 185
5.5 PA_FifoMsg Struct Reference 186
5.5.1 Detailed Description 187
5.6 PA_Point Struct Reference 188
5.6.1 Detailed Description 188
5.7 PA_TransferRegion Struct Reference 189
5.7.1 Detailed Description 189
5.8 PA::Point Class Reference 190
5.8.1 Detailed Description 190
5.9 SoundInfo Struct Reference 191
5.9.1 Detailed Description 191
5.10 PA::Sprite Class Reference 192
5.10.1 Detailed Description 193

6 Example Documentation **195**

6.1 Backgrounds/Effects/Mode7/source/main.c 195
6.2 Text/Normal/HelloWorld/source/main.c 197

Chapter 1

PAlib 0911XX Documentation

1.1 Introduction

Welcome to the PAlib documentation. Here you'll find information on how to use PAlib.

1.2 Core library

- **General functions** (p. 65)
- **Debugging utilities** (p. 47)
- **Math functions** (p. 83)
- **C++ wrappers** (p. 170)

1.3 Tiled backgrounds

- **Normal background functions** (p. 27)
- **Rotating background functions** (p. 24)
- **Dual background functions** (p. 153)
- **Text system** (p. 144)
- **Mode 7 functions** (p. 88)

1.4 Bitmapped backgrounds

- **Bitmapped background functions** (p. 49)

- **16-color bitmapped background functions** (p. 7)
- **Fake 16-bit background functions** (p. 60)
- **GIF functions** (p. 71)

1.5 Sprites

- **Sprite functions** (p. 105)
- **Dual sprite functions** (p. 130)
- **3D Sprite functions** (p. 13)

1.6 Palettes

- **Palette functions** (p. 92)
- **Dual palette functions** (p. 98)

1.7 Input

- **Pad and stylus functions** (p. 78)
- **Keyboard functions** (p. 74)
- **Handwriting recognition functions** (p. 101)
- **Microphone functions** (p. 87)
- **Special controller functions** (p. 82)

1.8 Sound

- **ASlib library** (p. 171)

1.9 Misc.

- **Special effects** (p. 102)

Chapter 2

Deprecated List

Global PA_16cCustomFont (p. **8**)

Global PA_8bitCustomFont (p. **147**)

Global PA_DualEasyBgLoad (p. **160**)

Global PA_DualLoadBg (p. **157**)

Global PA_DualLoadLargeBg (p. **158**)

Global PA_DualLoadLargeBgEx (p. **159**)

Global PA_DualLoadPAGfxLargeBg (p. **158**)

Global PA_DualLoadRotBg (p. **157**)

Global PA_DualLoadSimpleBg (p. **156**)

Global PA_DualLoadTiledBg (p. **155**)

Global PA_EasyBgLoad (p. **34**)

Global PA_EasyBgLoadPtr (p. **34**)

Global PA_InitCustomKeyboard (p. **75**)

Global PA_InitCustomText (p. **146**)

Global PA_LegacyIPCInit (p. **67**)

Global PA_LoadBg (p. **32**)

Global PA_LoadBgTiles (p. **31**)

Global PA_LoadLargeBg (p. **20**)

Global PA_LoadLargeBgEx (p. **21**)

Global PA_LoadPAGfxLargeBg (p. **20**)

Global PA_LoadPAGfxRotBg (p. **25**)

Global PA_LoadRotBg (p. **24**)

Global PA_LoadSimpleBg (p. **32**)

Global PA_LoadTiledBg (p. **31**)

Chapter 3

Module Documentation

3.1 16color pseudo-bitmap mode

Defines

- #define **PA_16cCustomFont**(c16_slot, c16_font)

  [DEPRECATED] Add custom fonts to the 16cText system !! Font must be converted with PAGfx

Functions

- static void **PA_Init16cBg** (u8 screen, u8 bg)

  Initialise 16color background on which you can paste images...

- void **PA_16cErase** (u8 screen)

  Erase the 16color background. Must be used right after PA_WaitForVBL to avoid glitches.

- static void **PA_Dual16cErase** (void)

  Erase the 16color background on both screens. Must be used right after PA_WaitForVBL to avoid glitches.

- static void **PA_InitComplete16c** (u8 bg, void ∗Palette)

  Initialise a 16color background on each screen and give them a given palette.

- s16 **PA_16cText** (u8 screen, s16 basex, s16 basey, s16 maxx, s16 maxy, const char ∗text, u8 color, u8 size, s32 limit)

  This is a variable width and variable size function to draw text on the screen.

- void **PA_Add16cFont** (int slot, const **PA_BgStruct** ∗font)

  Add a custom font to the 16c font system.

- ALWAYSINLINE void **PA_16cPutPixel** (u8 screen, s16 x, s16 y, u32 color)
  Plot a pixel on a 16c background.
- ALWAYSINLINE void **PA_16c8X4** (u8 screen, s16 x, s16 y, u32 *image)
  Load an 8x4 pixels image at a given position. Fastest of all pasting functions.
- ALWAYSINLINE void **PA_16c8X6** (u8 screen, s16 x, s16 y, u32 *image)
  Load an 8x6 pixels image at a given position. Second fastest of all pasting functions.
- ALWAYSINLINE void **PA_16c8X8** (u8 screen, s16 x, s16 y, u32 *image)
  Load an 8x8 pixels image at a given position.
- ALWAYSINLINE void **PA_16c8Xi** (u8 screen, s16 x, s16 y, u32 *image, u8 i)
  Load an 8xi row from a 8x16 pixels image at a given position. If i>16 the image is repeated.
- void **PA_16cClearZone** (u8 screen, s16 x1, s16 y1, s16 x2, s16 y2)
  Erase a 16c background zone.
- static u8 **PA_16cGetPixel** (u8 screen, s16 x, s16 y)
  Returns the pixel value of a given point on a 16c background.

3.1.1 Detailed Description

Special 16color background on which you can paste images. Usefull to show shots in SHMUP !

3.1.2 Define Documentation

3.1.2.1 #define PA_16cCustomFont(c16_slot, c16_font)

Value:

```
do{\
    PA_DEPRECATED_MACRO;\
    bittext_maps[c16_slot] = (u16*)(void*)c16_font##_Map;\
    c16_tiles[c16_slot] = (u32*)(void*)c16_font##_Tiles;\
    pa_bittextdefaultsize[c16_slot] = (u8*)c16_font##_Sizes;\
    pa_bittextpoliceheight[c16_slot] = c16_font##_Height;\
}while(0)
```

[DEPRECATED] Add custom fonts to the 16cText system !! Font must be converted with PAGfx

Deprecated

Parameters:

c16_slot Font slot... 0-4 are used by the defaut PAlib fonts, 5-9 are free to use. You can freely overwrite the PAlib fonts if you want

c16_font Font name...

3.1.3 Function Documentation

3.1.3.1 static inline void PA_Init16cBg (u8 *screen*, u8 *bg*) [inline, static]

Initialise 16color background on which you can paste images... Initialise 16color background on which you can paste images... Using palette 0.

Parameters:

screen Choose de screen (0 or 1)

bg Background number (0-3) Background number (0-3)

3.1.3.2 static inline void PA_16cErase (u8 *screen*)

Erase the 16color background. Must be used right after PA_WaitForVBL to avoid glitches.

Parameters:

screen Choose de screen (0 or 1)

3.1.3.3 static inline void PA_InitComplete16c (u8 *bg*, void * *Palette*) [inline, static]

Initialise a 16color background on each screen and give them a given palette.

Parameters:

bg Background number

Palette 16 color palette...

3.1.3.4 s16 PA_16cText (u8 *screen*, s16 *basex*, s16 *basey*, s16 *maxx*, s16 *maxy*, const char * *text*, u8 *color*, u8 *size*, s32 *limit*)

This is a variable width and variable size function to draw text on the screen.

Parameters:

screen Chose de screen (0 or 1)

basex X coordinate of the top left corner

basey Y coordinate of the top left corner

maxx X coordinate of the down right corner

maxy Y coordinate of the down right corner

text Text, such as "Hello World"

color Palette color to use (0-255)

size Size of the text, from 0 (really small) to 4 (pretty big)

limit You can give a maximum number of characters to output. This can be usefull to have a slowing drawing text (allow to draw 1 more character each frame...)

3.1.3.5 void PA_Add16cFont (int *slot*, const PA_BgStruct * *font*)

Add a custom font to the 16c font system.

Parameters:

slot Font slot. 0-4 are used by the default PAlib fonts, 5-9 are free to use. You can freely overwrite the PAlib fonts if you want.

font Pointer to the font.

3.1.3.6 ALWAYSINLINE PA_16cPutPixel (u8 *screen*, s16 *x*, s16 *y*, u32 *color*)

Plot a pixel on a 16c background.

Parameters:

screen Screen...

x X position in pixels of the top left corner. Note that it ranges from -8 to 263, in order to allow half-way offscreen images. NEVER DRAW BEYOND THESE LIMITS, or else you'll get major background glitches

y y position in pixels of the top left corner. Note that it ranges from -8 to 199, in order to allow half-way offscreen images. NEVER DRAW BEYOND THESE LIMITS, or else you'll get major background glitches

color Pixel value (0-15, uses the loaded palette)

3.1.3.7 ALWAYSINLINE void PA_16c8X4 (u8 *screen*, s16 *x*, s16 *y*, u32 * *image*)

Load an 8x4 pixels image at a given position. Fastest of all pasting functions.

Parameters:

screen Screen...

x X position in pixels of the top left corner. Note that it ranges from -8 to 255, in order to allow half-way offscreen images. NEVER DRAW BEYOND THESE LIMITS, or else you'll get major background glitches

y y position in pixels of the top left corner. Note that it ranges from -8 to 191, in order to allow half-way offscreen images. NEVER DRAW BEYOND THESE LIMITS, or else you'll get major background glitches

image 16 color image to load. Use (u32∗)ImageName if you get an error...

3.1.3.8 ALWAYSINLINE void PA_16c8X6 (u8 *screen*, s16 *x*, s16 *y*, u32 ∗ *image*)

Load an 8x6 pixels image at a given position. Second fastest of all pasting functions.

Parameters:

screen Screen...

x X position in pixels of the top left corner. Note that it ranges from -8 to 255, in order to allow half-way offscreen images. NEVER DRAW BEYOND THESE LIMITS, or else you'll get major background glitches

y y position in pixels of the top left corner. Note that it ranges from -8 to 191, in order to allow half-way offscreen images. NEVER DRAW BEYOND THESE LIMITS, or else you'll get major background glitches

image 16 color image to load. Use (u32∗)ImageName if you get an error...

3.1.3.9 ALWAYSINLINE void PA_16c8X8 (u8 *screen*, s16 *x*, s16 *y*, u32 ∗ *image*)

Load an 8x8 pixels image at a given position.

Parameters:

screen Screen...

x X position in pixels of the top left corner. Note that it ranges from -8 to 255, in order to allow half-way offscreen images. NEVER DRAW BEYOND THESE LIMITS, or else you'll get major background glitches

y y position in pixels of the top left corner. Note that it ranges from -8 to 191, in order to allow half-way offscreen images. NEVER DRAW BEYOND THESE LIMITS, or else you'll get major background glitches

image 16 color image to load. Use (u32∗)ImageName if you get an error...

3.1.3.10 ALWAYSINLINE void PA_16c8Xi (u8 *screen*, s16 *x*, s16 *y*, u32 ∗ *image*, u8 *i*)

Load an 8xi row from a 8x16 pixels image at a given position. If i>16 the image is repeated.

Parameters:

screen Screen...

x X position in pixels of the top left corner. Note that it ranges from -8 to 255, in order to allow half-way offscreen images. NEVER DRAW BEYOND THESE LIMITS, or else you'll get major background glitches

y y position in pixels of the top left corner. Note that it ranges from -8 to 191, in order to allow half-way offscreen images. NEVER DRAW BEYOND THESE LIMITS, or else you'll get major background glitches

image 16 color image to load. Use (u32∗)ImageName if you get an error...

i Number of lines of the image drawn (if greater than 16 the image will be repeated).

3.1.3.11 void PA_16cClearZone (u8 *screen*, s16 *x1*, s16 *y1*, s16 *x2*, s16 *y2*)

Erase a 16c background zone.

Parameters:

screen Screen...

x1 Upper left corner...

y1 Upper left corner...

x2 Lower right corner...

y2 Lower right corner...

3.1.3.12 static inline u8 PA_16cGetPixel (u8 *screen*, s16 *x*, s16 *y*) [inline, static]

Returns the pixel value of a given point on a 16c background.

Parameters:

screen Screen...

x X value...

y Y value...

3.2 3D Sprite System

Functions

- void **PA_Init3D** ()

  Initializes 3D.

- void **PA_Init3D2Banks** ()

  Initializes 3D taking two banks of VRAM.

- void **PA_3DProcess** ()

  Renders the 3D sprites.

- s16 **PA_3DCreateTex** (void ∗obj_data, u16 width, u16 height, u8 type)

  Creates a 3D texture.

- void **PA_3DCreateSpriteFromTex** (u16 sprite, u16 texture, u16 width, u16 height, u8 palette, s16 x, s16 y)

  Creates a 3D sprite from a texture.

- void **PA_Reset3DSprites** ()

  Resets the 3D system.

- void **PA_Reset3DSprites2Banks** ()

  Resets the dual bank 3D system.

- static void **PA_3DCreateSprite** (u16 sprite, void ∗image, u16 width, u16 height, u8 type, u8 palette, s16 x, s16 y)

  Creates a 3D sprite.

- void **PA_3DDeleteTex** (u32 tex_gfx)

  Deletes a 3D texture.

- static void **PA_3DDeleteSprite** (u16 sprite)

  Deletes a 3D sprite.

- static void **PA_3DSetSpriteX** (u16 sprite, s16 x)

  Moves a 3D sprite in the X axis.

- static void **PA_3DSetSpriteY** (u16 sprite, s16 y)

  Moves a 3D sprite in the Y axis.

- static void **PA_3DSetSpriteXY** (u16 sprite, s16 x, s16 y)

  Moves a 3D sprite.

- static void **PA_3DSetSpriteRotateX** (u16 sprite, s16 rotateX)

Rotates a 3D sprite in the X axis.

- static void **PA_3DSetSpriteRotateY** (u16 sprite, s16 rotateY)

  Rotates a 3D sprite in the Y axis.

- static void **PA_3DSetSpriteRotateZ** (u16 sprite, s16 rotate)

  Rotates a 3D sprite in the Z axis.

- static void **PA_3DSetSpriteRotateXYZ** (u16 sprite, s16 rotateX, s16 rotateY, s16 rotateZ)

  Rotates a 3D sprite.

- static void **PA_3DSetSpriteZoomX** (u16 sprite, float zoomx)

  Zooms a 3D sprite horizontally.

- static void **PA_3DSetSpriteZoomY** (u16 sprite, float zoomy)

  Zooms a 3D sprite vertically.

- static void **PA_3DSetSpriteZoomXY** (u16 sprite, float zoomx, float zoomy)

  Zooms a 3D sprite.

- static void **PA_3DSetSpriteWidth** (u16 sprite, u16 width)

  Changes the width of a 3D sprite.

- static void **PA_3DSetSpriteHeight** (u16 sprite, u16 height)

  Changes the height of a 3D sprite.

- static void **PA_3DSetSpriteWidthHeight** (u16 sprite, u16 width, u16 height)

  Changes the size of a 3D sprite.

- static void **PA_3DSetSpriteHflip** (u16 sprite, u8 hflip)

  Sets the HFlip of a 3D sprite.

- static void **PA_3DSetSpriteVflip** (u16 sprite, u8 vflip)

  Sets the VFlip of a 3D sprite.

- static u8 **PA_3DSpriteTouched** (u16 sprite)

  Retrives if a 3D sprite is being touched by the stylus.

- static void **PA_3DSetSpriteTex** (u16 sprite, u16 texture)

  Sets the texture of a 3D sprite.

- static void **PA_3DSetSpritePal** (u16 sprite, u16 palette)

  Sets the palette of a 3D sprite.

- void **PA_3DSetSpriteFrame** (u16 sprite, u16 frame)

Sets the animation frame of a 3D sprite.

- static void **PA_3DSetSpriteTopLeft** (u16 sprite, s16 x, s16 y)

  Sets the top left corner of a 3D sprite.

- static void **PA_3DSetSpriteTopRight** (u16 sprite, s16 x, s16 y)

  Sets the top right corner of a 3D sprite.

- static void **PA_3DSetSpriteBottomLeft** (u16 sprite, s16 x, s16 y)

  Sets the bottom left corner of a 3D sprite.

- static void **PA_3DSetSpriteBottomRight** (u16 sprite, s16 x, s16 y)

  Sets the bottom right corner of a 3D sprite.

- static void **PA_3DSetSpritePrio** (u16 sprite, u16 priority)

  Sets the priority of a 3D sprite.

- static void **PA_3DSetSpritePolyID** (u16 sprite, u8 polyID)

  Sets the PolyID of a 3D sprite.

- static void **PA_3DSetSpriteAlpha** (u16 sprite, u8 alpha)

  Sets the alpha value of a 3D sprite.

- void **PA_3DStartSpriteAnimEx** (u16 sprite, s16 firstframe, s16 lastframe, s16 speed, u8 type, s16 ncycles)

  Start a 3D sprite animation. Once started, it continues on and on by itself until you stop it !

- static void **PA_3DStartSpriteAnim** (u16 sprite, s16 firstframe, s16 lastframe, s16 speed)

  Start a sprite animation. Once started, it continues on and on by itself until you stop it !

- static void **PA_3DStopSpriteAnim** (u16 sprite)

  Stop a sprite animation.

- static void **PA_3DSetSpriteAnimFrame** (u16 sprite, u16 frame)

  Set the current animation frame number.

- static u16 **PA_3DGetSpriteAnimFrame** (u16 sprite)

  Returns the current animation frame number.

- static void **PA_3DSetSpriteAnimSpeed** (u16 sprite, s16 speed)

  Set the current animation speed.

- static u16 **PA_3DGetSpriteAnimSpeed** (u16 sprite)

  Returns the current animation speed.

- static void **PA_3DSetSpriteNCycles** (u16 sprite, s16 NCycles)

  Set the current animation cycles left (-1 for inifinite loop).

- static u16 **PA_3DGetSpriteNCycles** (u16 sprite)

  Returns the current number of animation cycles left.

- static void **PA_3DSpriteAnimPause** (u16 sprite, u8 pause)

  Pause or UnPause a sprite animation.

- static s32 **PA_3DGetSpriteX** (u16 sprite)

  Gets the X value of a 3D sprite.

- static s32 **PA_3DGetSpriteY** (u16 sprite)

  Gets the Y value of a 3D sprite.

- static void **PA_3DSetSpriteVisible** (u16 sprite, u8 visible)

  Retrieves if a 3D sprite is visible.

3.2.1 Detailed Description

Sprites on one screen using the DS's 3D GPU

3.2.2 Function Documentation

3.2.2.1 void PA_3DStartSpriteAnimEx (u16 *sprite*, s16 *firstframe*, s16 *lastframe*, s16 *speed*, u8 *type*, s16 *ncycles*)

Start a 3D sprite animation. Once started, it continues on and on by itself until you stop it !

Parameters:

sprite sprite number in the sprite system

firstframe First frame of the animation sequence, most of the time 0...

lastframe Last frame to be displayed. When it gets there, it loops back to the first frame

speed Speed, in frames per second. So speed 1 would mean 1 image per second, so 1 image every game frame

type Defines how you want it to loop. ANIM_LOOP (0) for a normal loop, ANIM_UPDOWN (1) for back and forth animation.

ncycles Number of animation cycles before stopping. If using ANIM_UPDOWN, it takes 2 cycles to come back to the original image

3.2.2.2 static inline void PA_3DStartSpriteAnim (u16 *sprite*, s16 *firstframe*, s16 *lastframe*, s16 *speed*) [inline, static]

Start a sprite animation. Once started, it continues on and on by itself until you stop it !

Parameters:

sprite sprite number in the sprite system

firstframe First frame of the animation sequence, most of the time 0...

lastframe Last frame to be displayed. When it gets there, it loops back to the first frame

speed Speed, in frames per second. So speed 1 would mean 1 image per second, so 1 image every game frame

3.2.2.3 static inline void PA_3DStopSpriteAnim (u16 *sprite*) [inline, static]

Stop a sprite animation.

Parameters:

sprite sprite number in the sprite system

3.2.2.4 static inline void PA_3DSetSpriteAnimFrame (u16 *sprite*, u16 *frame*) [inline, static]

Set the current animation frame number.

Parameters:

sprite sprite number in the sprite system

frame Frame number to use...

3.2.2.5 static inline u16 PA_3DGetSpriteAnimFrame (u16 *sprite*) [inline, static]

Returns the current animation frame number.

Parameters:

sprite sprite number in the sprite system

3.2.2.6 static inline void PA_3DSetSpriteAnimSpeed (u16 *sprite*, s16 *speed*) [inline, static]

Set the current animation speed.

Parameters:

sprite sprite number in the sprite system

speed Speed, in fps...

3.2.2.7 static inline u16 PA_3DGetSpriteAnimSpeed (u16 *sprite*) [inline, static]

Returns the current animation speed.

Parameters:

sprite sprite number in the sprite system

3.2.2.8 static inline void PA_3DSetSpriteNCycles (u16 *sprite*, s16 *NCycles*) [inline, static]

Set the current animation cycles left (-1 for inifinite loop).

Parameters:

sprite sprite number in the sprite system

NCycles Number of cycles

3.2.2.9 static inline u16 PA_3DGetSpriteNCycles (u16 *sprite*) [inline, static]

Returns the current number of animation cycles left.

Parameters:

sprite sprite number in the sprite system

3.2.2.10 static inline u16 PA_3DSpriteAnimPause (u16 *sprite*, u8 *pause*) [inline, static]

Pause or UnPause a sprite animation.

Parameters:

sprite sprite number in the sprite system

pause 1 for pause, 0 for unpause

3.3 Old large background system

Defines

- #define **PA_LoadLargeBg**(screen, bg_select, bg_tiles, bg_map, color_mode, lx, ly)

  [DEPRECATED] Completely load and initialise a background with infinite scrolling (usefull if larger or wider than 512 pixels)

- #define **PA_LoadPAGfxLargeBg**(screen, bg_number, bg_name)

  [DEPRECATED] Completely load and initialise a background with infinite scrolling (usefull if larger or wider than 512 pixels), converted with PAGfx

- #define **PA_LoadLargeBgEx**(screen, bg_select, bg_tiles, tile_size, bg_map, color_mode, lx, ly)

  [DEPRECATED] Completely load and initialise a background with infinite scrolling (usefull if larger or wider than 512 pixels), but here you can put yourself the tile size...

Functions

- static void **PA_InfLargeScrollX** (u8 screen, u8 bg_select, s32 x)

  Scroll a large infinite scrolling background horizontaly. It must have been initialised with PA_LoadLargeBg.

- static void **PA_InfLargeScrollY** (u8 screen, u8 bg_select, s32 y)

  Scroll a large infinite scrolling background vertically. It must have been initialised with PA_LoadLargeBg.

- static void **PA_InfLargeScrollXY** (u8 screen, u8 bg_select, s32 x, s32 y)

  Scroll a large infinite scrolling background horizontaly and vertically. It must have been initialised with PA_LoadLargeBg.

- static void **PA_LargeScrollX** (u8 screen, u8 bg_select, s32 x)

  Scroll a large background horizontaly. It must have been initialised with PA_LoadLargeBg. This function does not wrap around, but is faster than the InfLargeScroll...

- static void **PA_LargeScrollY** (u8 screen, u8 bg_select, s32 y)

  Scroll a large background vertically. It must have been initialised with PA_LoadLargeBg. This function does not wrap around, but is faster than the InfLargeScroll...

- static void **PA_LargeScrollXY** (u8 screen, u8 bg_select, s32 x, s32 y)

  Scroll a large background horizontaly and vertically. It must have been initialised with PA_LoadLargeBg. This function does not wrap around, but is faster than the InfLargeScroll...

3.3.1 Detailed Description

Old LargeMap functions, obsoleted by **PA_LoadBackground()** (p. 37)

3.3.2 Define Documentation

3.3.2.1 #define PA_LoadLargeBg(screen, bg_select, bg_tiles, bg_map, color_mode, lx, ly)

Value:

```
do{\
    PA_DEPRECATED_MACRO;\
    PA_BgInfo[screen][bg_select].NTiles = SIZEOF_16BIT(bg_tiles)>>5;\
    if (PA_BgInfo[screen][bg_select].NTiles < MAX_TILES) {PA_LoadSimpleBg(screen,
       bg_select, bg_tiles, NULL, BG_512X256, 0, color_mode);}\
    else{PA_LoadTileEngine(screen, bg_select, (void*)bg_tiles);}\
    PA_InitLargeBg(screen, bg_select, lx, ly, (void*)bg_map);}while(0)
```

[DEPRECATED] Completely load and initialise a background with infinite scrolling (usefull if larger or wider than 512 pixels)

Deprecated

Parameters:

screen Chose de screen (0 or 1)

bg_select Background number to load (from 0 to 3)

bg_tiles Name of the tiles' info (example: ship_Tiles)

bg_map Name of the map's info (example : ship_Map)

color_mode Color mode : 0 for 16 color mode, 1 for 256...

lx Width, in tiles. So a 512 pixel wide map is 64 tiles wide...

ly Height, in tiles. So a 512 pixel high map is 64 tiles high...

3.3.2.2 #define PA_LoadPAGfxLargeBg(screen, bg_number, bg_name)

Value:

```
do{\
    PA_DEPRECATED_MACRO;\
    PA_LoadBgPal(screen, bg_number, (void*)bg_name##_Pal); \
    PA_LoadLargeBg(screen, bg_number, bg_name##_Tiles, bg_name##_Map, 1, (bg_name
      ##_Info[1]) >> 3, (bg_name##_Info[2]) >> 3);}while(0)
```

[DEPRECATED] Completely load and initialise a background with infinite scrolling (usefull if larger or wider than 512 pixels), converted with PAGfx

Deprecated

Parameters:

screen Chose de screen (0 or 1)

bg_number Background number to load (from 0 to 3)

bg_name Background name, in PAGfx

3.3.2.3 #define PA_LoadLargeBgEx(screen, bg_select, bg_tiles, tile_size, bg_map, color_mode, lx, ly)

Value:

```
do{\
    PA_DEPRECATED_MACRO;\
    PA_BgInfo[screen][bg_select].NTiles = SIZEOF_16BIT(bg_tiles)>>5;\
    if (PA_BgInfo[screen][bg_select].NTiles < MAX_TILES) {PA_LoadBg(screen, bg_se
      lect, bg_tiles, tile_size, NULL, BG_512X256, 0, color_mode);}\
    else{PA_LoadTileEngine(screen, bg_select, bg_tiles);}\
    PA_InitLargeBg(screen, bg_select, lx, ly, (void*)bg_map);}while(0)
```

[DEPRECATED] Completely load and initialise a background with infinite scrolling (usefull if larger or wider than 512 pixels), but here you can put yourself the tile size...

Deprecated

Parameters:

screen Chose de screen (0 or 1)

bg_select Background number to load (from 0 to 3)

bg_tiles Name of the tiles' info (example: ship_Tiles)

tile_size Size of your tileset

bg_map Name of the map's info (example : ship_Map)

color_mode Color mode : 0 for 16 color mode, 1 for 256...

lx Width, in tiles. So a 512 pixel wide map is 64 tiles wide...

ly Height, in tiles. So a 512 pixel high map is 64 tiles high...

3.3.3 Function Documentation

3.3.3.1 void PA_InfLargeScrollX (u8 *screen*, u8 *bg_select*, s32 *x*) [inline, static]

Scroll a large infinite scrolling background horizontaly. It must have been initialised with PA_LoadLargeBg.

Parameters:

screen Chose de screen (0 or 1)

bg_select Background number to load (from 0 to 3)

x X value to scroll

3.3.3.2 void PA_InfLargeScrollY (u8 *screen*, u8 *bg_select*, s32 *y*) [inline, static]

Scroll a large infinite scrolling background vertically. It must have been initialised with PA_LoadLargeBg.

Parameters:

screen Chose de screen (0 or 1)

bg_select Background number to load (from 0 to 3)

y Y value to scroll

3.3.3.3 static inline void PA_InfLargeScrollXY (u8 *screen*, u8 *bg_select*, s32 *x*, s32 *y*) [inline, static]

Scroll a large infinite scrolling background horizontaly and vertically. It must have been initialised with PA_LoadLargeBg.

Parameters:

screen Chose de screen (0 or 1)

bg_select Background number to load (from 0 to 3)

x X value to scroll

y Y value to scroll

3.3.3.4 void PA_LargeScrollX (u8 *screen*, u8 *bg_select*, s32 *x*) [inline, static]

Scroll a large background horizontaly. It must have been initialised with PA_LoadLargeBg. This function does not wrap around, but is faster than the InfLargeScroll...

Parameters:

screen Chose de screen (0 or 1)

bg_select Background number to load (from 0 to 3)

x X value to scroll

3.3.3.5 void PA_LargeScrollY (u8 *screen*, u8 *bg_select*, s32 *y*) [inline, static]

Scroll a large background vertically. It must have been initialised with PA_LoadLargeBg. This function does not wrap around, but is faster than the InfLargeScroll...

Parameters:

screen Chose de screen (0 or 1)

bg_select Background number to load (from 0 to 3)

y Y value to scroll

3.3.3.6 static inline void PA_LargeScrollXY (u8 *screen*, u8 *bg_select*, s32 *x*, s32 *y*) [inline, static]

Scroll a large background horizontaly and vertically. It must have been initialised with PA_LoadLargeBg. This function does not wrap around, but is faster than the InfLargeScroll...

Parameters:

screen Chose de screen (0 or 1)

bg_select Background number to load (from 0 to 3)

x X value to scroll

y Y value to scroll

3.4 Rotating Backgrounds

Defines

- #define **PA_LoadRotBg**(screen, bg_select, bg_tiles, bg_map, bg_size, wraparound)

  [DEPRECATED] Load a background fit for rotating/scaling! Warning, you must use PA_SetVideoMode to 1 if you want 1 rotating background (Bg3 only !), or 2 for 2 rotating backgrounds (Bg2 and 3). The background MUST be in 256 colors

- #define **PA_LoadPAGfxRotBg**(screen, bg_select, bg_name, wraparound)

  [DEPRECATED] Load a background fit for rotating/scaling ! Warning, you must use PA_SetVideoMode to 1 if you want 1 rotating background (Bg3 only !), or 2 for 2 rotating backgrounds (Bg2 and 3). The background MUST be in 256 colors

Functions

- static void **PA_SetBgRot** (u8 screen, u8 bg_select, s32 x_scroll, s32 y_scroll, s32 x_rotcentre, s32 y_rotcentre, s16 bg_angle, s32 bg_zoom)

  Rotate/Scale a RotBg.

3.4.1 Detailed Description

Load rotating backgrounds, move, rotate, scale them

3.4.2 Define Documentation

3.4.2.1 #define PA_LoadRotBg(screen, bg_select, bg_tiles, bg_map, bg_size, wraparound)

Value:

```
do{\
    PA_DEPRECATED_MACRO;\
    PA_DeleteBg(screen, bg_select);\
    PA_LoadBgTiles(screen, bg_select, bg_tiles); \
    PA_LoadRotBgMap(screen, bg_select, (void*)bg_map, bg_size); \
    PA_InitBg(screen, bg_select, bg_size, wraparound, 1);\
    PA_SetBgRot(screen, bg_select, 0, 0, 0, 0, 0, 256);\
    }while(0)
```

[DEPRECATED] Load a background fit for rotating/scaling! Warning, you must use PA_SetVideoMode to 1 if you want 1 rotating background (Bg3 only !), or 2 for 2 rotating backgrounds (Bg2 and 3). The background MUST be in 256 colors

Deprecated

Parameters:

screen Chose de screen (0 or 1)

bg_select Background number to load

bg_tiles Name of the tiles' info (example: ship_Tiles)

bg_map Name of the map's info (example : ship_Map)

bg_size Background size. Use the following macros : BG_ROT_128X128, or 256X256, 512X512, or 1024X1024

wraparound If the background wraps around or not.

3.4.2.2 #define PA_LoadPAGfxRotBg(screen, bg_select, bg_name, wraparound)

Value:

```
do{\
    PA_DEPRECATED_MACRO;\
    PA_Load8bitBgPal(screen, (void*)bg_name##_Pal);\
    PA_LoadRotBg(screen, bg_select, bg_name##_Tiles, bg_name##_Map, PA_GetPAGfxRo
      tBgSize(bg_name##_Info[1]), wraparound);\
    }while(0)
```

[DEPRECATED] Load a background fit for rotating/scaling ! Warning, you must use PA_SetVideoMode to 1 if you want 1 rotating background (Bg3 only !), or 2 for 2 rotating backgrounds (Bg2 and 3). The background MUST be in 256 colors

Deprecated

Parameters:

screen Chose de screen (0 or 1)

bg_select Background number to load

bg_name Background name, like bg0

wraparound If the background wraps around or not.

3.4.3 Function Documentation

3.4.3.1 static inline void PA_SetBgRot (u8 *screen*, u8 *bg_select*, s32 *x_scroll*, s32 *y_scroll*, s32 *x_rotcentre*, s32 *y_rotcentre*, s16 *bg_angle*, s32 *bg_zoom*) [inline, static]

Rotate/Scale a RotBg.

Parameters:

screen Chose de screen (0 or 1)

bg_select Background number to load

x_scroll X Scroll...

y_scroll Y Scroll...

x_rotcentre X position of the rotation center

y_rotcentre Y position of the rotation center

bg_angle Rotation Angle (0-511)

bg_zoom Zoom (256 for no zoom)

3.5 Tiled Background Modes

Data Structures

- struct **PA_BgStruct**

  Background structure.

Defines

- #define **_GFX_ALIGN** __attribute__((aligned (4)))

  Graphics align define for PAGfx.

- #define **PA_HideBg**(screen, bg_select) _REG16(REG_BGSCREEN(screen)) &= ∼(0x100 << (bg_select))

  Hide a screen's background.

- #define **PA_ShowBg**(screen, bg_select) _REG16(REG_BGSCREEN(screen)) |= (0x100 << (bg_select))

  Show a hidden background.

- #define **PA_ResetBg**(screen) _REG16(REG_BGSCREEN(screen)) &= ∼(0xF00)

  Reinitialize de Bg system of a screen. It only hides all the backgrounds in reality...

- #define **PA_LoadBgTiles**(screen, bg_select, bg_tiles) PA_-LoadBgTilesEx(screen, bg_select, (void∗)bg_tiles, SIZEOF_16BIT(bg_tiles))

  [DEPRECATED] Load a tileset into memory

- #define **PA_LoadTiledBg**(screen, bg_number, bg_name)

  [DEPRECATED] This will never get easier... Loads a background TiledBg converted with PAGfx, with its tiles, map, and palette. Only 256 color mode available.

- #define **PA_LoadSimpleBg**(screen, bg_select, bg_tiles, bg_map, bg_size, wraparound, color_mode)

  [DEPRECATED] Simple way to load a Background. Combines PA_InitBg, PA_-LoadBgTiles, and PA_LoadBgMap

- #define **PA_LoadBg**(screen, bg_select, bg_tiles, tile_size, bg_map, bg_size, wraparound, color_mode)

  [DEPRECATED] Simplest way to load a Background. Combines PA_InitBg, PA_-LoadBgTiles, and PA_LoadBgMap

- #define **PA_SetMapTileAll**(screen, bg_select, x, y, tile_info) ∗(u16∗)(PA_-BgInfo[screen][bg_select].Map + ((x) << 1) + ((y) << 6)) = (tile_info)

  Change the tile info used by a given tile in the map.

- #define **PA_EasyBgLoad**(screen, bg_number, bg_name)

  [DEPRECATED] Easiest way to load a background converted with PAGfx...

- #define **PA_EasyBgLoadPtr**(screen, bg_number, bg_name)

  [DEPRECATED] Easiest way to load a background converted with PAGfx... Can take pointers !

Enumerations

- enum {

  PA_BgInvalid, **PA_BgNormal**, **PA_BgLarge**, **PA_BgUnlimited**,

  PA_BgRot, **PA_Font1bit**, **PA_Font4bit**, **PA_Font8bit** }

  Types of background.

Functions

- void **PA_ResetBgSys** (void)

  Reset the background system.

- void **PA_ResetBgSysScreen** (u8 screen)

  Reset the background system on 1 screen.

- void **PA_InitBg** (u8 screen, u8 bg_select, u8 bg_size, u8 wraparound, u8 color_mode)

  Initialise a given background. Do this only after having loaded a tileset and a map.

- void **PA_LoadBgTilesEx** (u8 screen, u8 bg_select, void ∗bg_tiles, u32 size)

  Load a tileset into memory with a given size.

- void **PA_ReLoadBgTiles** (u8 screen, u8 bg_select, void ∗bg_tiles)

  ReLoad a tileset into memory.

- void **PA_DeleteTiles** (u8 screen, u8 bg_select)

  Delete a tilest in memory. Note that loading a tileset automatically deletes the preceding one, so you won't need to use this function often.

- void **PA_DeleteMap** (u8 screen, u8 bg_select)

  Delete a map in memory. Note that loading a map automatically deletes the preceding one, so you won't need to use this function often.

- static void **PA_DeleteBg** (u8 screen, u8 bg_select)

  Delete and reset a complete background.

- void **PA_LoadBgMap** (u8 screen, u8 bg_select, void *bg_map, u8 bg_size)

  Load a background's map info.

- void **PA_LoadBackground** (u8 screen, u8 bg_select, const **PA_BgStruct** *bg_name)

  Load a background (EasyBg or RotBg).

- static void **PA_BGScrollX** (u8 screen, u8 bg_number, s32 x)

  Scroll horizontaly a Tiled background.

- static void **PA_BGScrollY** (u8 screen, u8 bg_number, s32 y)

  Scroll vertically a Tiled background.

- static void **PA_SetMapTile** (u8 screen, u8 bg_select, s16 x, s16 y, s16 tile_number)

  Change the tile gfx used by a given tile in the map.

- static void **PA_SetLargeMapTile** (u8 screen, u8 bg_select, s32 x, s32 y, u32 tile_info)

  Change the tile info used by a given tile in the map, only for big background (512 large or wide).

- static void **PA_SetMapTileHflip** (u8 screen, u8 bg_select, u8 x, u8 y, u8 hflip)

  Flip a given tile horizontaly.

- static void **PA_SetMapTileVflip** (u8 screen, u8 bg_select, u8 x, u8 y, u8 vflip)

  Flip a given tile verticaly.

- static void **PA_SetMapTilePal** (u8 screen, u8 bg_select, u8 x, u8 y, u8 palette_number)

  Change the 16 color palette used by a tile. Works only if the Bg is in 16 colors...

- static void **PA_SetBgPrio** (u8 screen, u8 bg, u8 prio)

  Change a backgrounds priority.

- static void **PA_SetBgPrioSeq** (u8 screen, u8 priority0, u8 priority1, u8 priority2, u8 priority3)

  Change all the background priorities to a given background order.

- static void **PA_ClearBg** (u8 screen, u8 bg_select)

  Erase a given background (just the tilemap).

- void **PA_EasyBgScrollX** (u8 screen, u8 bg_number, s32 x)

  Scroll horizontaly any background.

- void **PA_EasyBgScrollY** (u8 screen, u8 bg_number, s32 y)

  Scroll vertically any background.

- static void **PA_EasyBgScrollXY** (u8 screen, u8 bg_number, s32 x, s32 y)
  Scroll horizontaly and vertically any background.
- static u8 **PA_EasyBgGetPixel** (u8 screen, u8 bg_number, s32 x, s32 y)
  Returns the color (number in the palette) of the screen pixel...
- static u16 **PA_EasyBgGetPixelCol** (u8 screen, u8 bg_number, s32 x, s32 y)
  Returns the color (u16 value) of the screen pixel...
- static void **PA_SetBgWrap** (u8 screen, u8 bg, u8 wrap)
  Set on/off the background wrapping (for rotating, 8bit, and 16bit backgrounds).
- static void **PA_InitParallaxX** (u8 screen, s32 bg0, s32 bg1, s32 bg2, s32 bg3)
  Initialise Parallax Scrolling for multiple backgrounds, horizontaly. Chose the speed at which each background will scroll compared to the others. Then use PA_ParallaxScrollX to scroll...
- static void **PA_InitParallaxY** (u8 screen, s32 bg0, s32 bg1, s32 bg2, s32 bg3)
  Initialise Parallax Scrolling for multiple backgrounds, horizontaly. Chose the speed at which each background will scroll compared to the others. Then use PA_ParallaxScrollX to scroll...
- static void **PA_ParallaxScrollX** (u8 screen, s32 x)
  Scroll the backgrounds.
- static void **PA_ParallaxScrollY** (u8 screen, s32 y)
  Scroll the backgrounds.
- static void **PA_ParallaxScrollXY** (u8 screen, s32 x, s32 y)
  Scroll the backgrounds.

3.5.1 Detailed Description

Load a background, scroll it, etc...

3.5.2 Define Documentation

3.5.2.1 #define PA_HideBg(screen, bg_select) _REG16(REG_BGSCREEN(screen)) &= ~(0x100 << (bg_select))

Hide a screen's background.

Parameters:

screen Choose the screen (0 or 1)

bg_select Background number to load (from 0 to 3)

3.5.2.2 #define PA_ShowBg(screen, bg_select) _REG16(REG_BGSCREEN(screen)) |= (0x100 << (bg_select))

Show a hidden background.

Parameters:

screen Choose the screen (0 or 1)

bg_select Background number to load (from 0 to 3)

3.5.2.3 #define PA_ResetBg(screen) _REG16(REG_BGSCREEN(screen)) &= ~(0xF00)

Reinitialize de Bg system of a screen. It only hides all the backgrounds in reality...

Parameters:

screen Choose the screen (0 or 1)

3.5.2.4 #define PA_LoadBgTiles(screen, bg_select, bg_tiles) PA_LoadBgTilesEx(screen, bg_select, (void*)bg_tiles, SIZEOF_16BIT(bg_tiles))

[DEPRECATED] Load a tileset into memory

Deprecated

Parameters:

screen Choose the screen (0 or 1)

bg_select Background number to load (from 0 to 3)

bg_tiles Name of the tiles' info (example: ship_Tiles)

3.5.2.5 #define PA_LoadTiledBg(screen, bg_number, bg_name)

Value:

```
do{\
    PA_DEPRECATED_MACRO;\
    PA_LoadBgPal(screen, bg_number, (void*)bg_name##_Pal); \
    PA_LoadSimpleBg(screen, bg_number, bg_name##_Tiles, bg_name##_Map, PA_GetPAGf
      xBgSize(bg_name##_Info[1], bg_name##_Info[2]), 0, 1);}while(0)
```

[DEPRECATED] This will never get easier... Loads a background TiledBg converted with PAGfx, with its tiles, map, and palette. Only 256 color mode available.

Deprecated

Parameters:

screen Choose the screen (0 or 1)

bg_number Background number to load (from 0 to 3)

bg_name Background name, like bg0

3.5.2.6 #define PA_LoadSimpleBg(screen, bg_select, bg_tiles, bg_map, bg_size, wraparound, color_mode)

Value:

```
do{\
    PA_DEPRECATED_MACRO;\
    PA_DeleteBg(screen, bg_select);\
    PA_LoadBgTiles(screen, bg_select, bg_tiles); \
    PA_LoadBgMap(screen, bg_select, (void*)bg_map, bg_size); \
    PA_InitBg(screen, bg_select, bg_size, 0, color_mode);\
    PA_BGScrollXY(screen, bg_select, 0, 0);}while(0)
```

[DEPRECATED] Simple way to load a Background. Combines PA_InitBg, PA_-LoadBgTiles, and PA_LoadBgMap

Deprecated

Parameters:

screen Choose the screen (0 or 1)

bg_select Background number to load (from 0 to 3)

bg_tiles Name of the tiles' info (example: ship_Tiles)

bg_map Name of the map's info (example : ship_Map)

bg_size Background size. To use a normal background, use the macros BG_-256X256, BG_256X512, etc...

wraparound If the background wraps around or not. More important for rotating backgrounds.

color_mode Color mode : 0 for 16 color mode, 1 for 256...

3.5.2.7 #define PA_LoadBg(screen, bg_select, bg_tiles, tile_size, bg_map, bg_size, wraparound, color_mode)

Value:

```
do{\
    PA_DEPRECATED_MACRO;\
    PA_LoadBgTilesEx(screen, bg_select, (void*)bg_tiles, tile_size); \
    PA_LoadBgMap(screen, bg_select, (void*)bg_map, bg_size); \
    PA_InitBg(screen, bg_select, bg_size, 0, color_mode);\
    PA_BGScrollXY(screen, bg_select, 0, 0);}while(0)
```

[DEPRECATED] Simplest way to load a Background. Combines PA_InitBg, PA_-LoadBgTiles, and PA_LoadBgMap

Deprecated

Parameters:

screen Choose the screen (0 or 1)

bg_select Background number to load (from 0 to 3)

bg_tiles Name of the tiles' info (example: ship_Tiles)

tile_size Size of your tileset

bg_map Name of the map's info (example : ship_Map)

bg_size Background size. This is important, because it also determines whether the Bg is rotatable or not. To use a normal background, use the macros BG_256X256, BG_256X512, etc... For a rotatable Bg, use the macros BG_-ROT_128X128...

wraparound If the background wraps around or not. More important for rotating backgrounds.

color_mode Color mode : 0 for 16 color mode, 1 for 256...

3.5.2.8 #define PA_SetMapTileAll(screen, bg_select, x, y, tile_info) ∗(u16∗)(PA_BgInfo[screen][bg_select].Map + ((x) << 1) + ((y) << 6)) = (tile_info)

Change the tile info used by a given tile in the map.

Parameters:

screen Choose the screen (0 or 1)

bg_select Background number (0-3)

x X value of the tile to change

y Y value of the map tile to change

tile_info New tile to put (tile + palette + flips...)

3.5.2.9 #define PA_EasyBgLoad(screen, bg_number, bg_name)

Value:

```
do{PA_BgInfo[screen][bg_number].BgMode = bg_name##_Info[0];\
    PA_DEPRECATED_MACRO;\
    PA_StoreEasyBgInfos(screen, bg_number, bg_name##_Info[0], bg_name##_Info[1],
      bg_name##_Info[2], (void*)bg_name##_Tiles, SIZEOF_16BIT(bg_name##_Tiles), (void*)
      bg_name##_Map, SIZEOF_16BIT(bg_name##_Map), (void*)bg_name##_Pal);\
    if(PA_BgInfo[screen][bg_number].BgMode == BG_TILEDBG){  PA_LoadTiledBg(screen
      , bg_number, bg_name);}\
    else{PA_LoadPAGfxLargeBg(screen, bg_number, bg_name);}}while(0)
```

[DEPRECATED] Easiest way to load a background converted with PAGfx...

Deprecated

Parameters:

screen Choose de screen (0 or 1)

bg_number Background number... (0-3)

bg_name Background name

3.5.2.10 #define PA_EasyBgLoadPtr(screen, bg_number, bg_name)

Value:

```
do{\
    PA_DEPRECATED_MACRO;\
    PA_EasyBgLoadEx(screen, bg_number, (u32*)bg_name->Info, bg_name->Tiles, bg_na
      me->TileSize, bg_name->Map, bg_name->MapSize, bg_name->Palette);\
}while(0)
```

[DEPRECATED] Easiest way to load a background converted with PAGfx... Can take pointers !

Deprecated

Parameters:

screen Choose de screen (0 or 1)

bg_number Background number... (0-3)

bg_name Background, like &bg0

3.5.3 Enumeration Type Documentation

3.5.3.1 anonymous enum

Types of background.

Enumerator:

PA_BgInvalid Invalid background.

PA_BgNormal Normal tiled background AKA TiledBg.

PA_BgLarge Large background AKA LargeMap.

PA_BgUnlimited Unlimited background AKA InfiniteMap.

PA_BgRot Rotational background.

PA_Font1bit 1-bit bitmap font

PA_Font4bit 4-bit bitmap font

PA_Font8bit 8-bit bitmap font

3.5.4 Function Documentation

3.5.4.1 void PA_ResetBgSysScreen (u8 *screen*)

Reset the background system on 1 screen.

Parameters:

screen Choose the screen (0 or 1)

3.5.4.2 void PA_InitBg (u8 *screen*, u8 *bg_select*, u8 *bg_size*, u8 *wraparound*, u8 *color_mode*)

Initialise a given background. Do this only after having loaded a tileset and a map.

Parameters:

screen Choose the screen (0 or 1)

bg_select Background number to load (from 0 to 3)

bg_size Background size. This is important, because it also determines whether the Bg is rotatable or not. To use a normal background, use the macros BG_256X256, BG_256X512, etc... For a rotatable Bg, use the macros BG_ROT_128X128...

wraparound If the background wraps around or not. More important for rotating backgrounds.

color_mode Color mode : 0 for 16 color mode, 1 for 256...

3.5.4.3 void PA_LoadBgTilesEx (u8 *screen*, u8 *bg_select*, void * *bg_tiles*, u32 *size*)

Load a tileset into memory with a given size.

Parameters:

screen Choose the screen (0 or 1)

bg_select Background number to load (from 0 to 3)

bg_tiles Name of the tiles' info (example: ship_Tiles)

size 16 bit size...

3.5.4.4 void PA_ReLoadBgTiles (u8 *screen*, u8 *bg_select*, void * *bg_tiles*)

ReLoad a tileset into memory.

Parameters:

screen Choose the screen (0 or 1)

bg_select Background number to load (from 0 to 3)

bg_tiles Name of the tiles' info (example: ship_Tiles)

3.5.4.5 void PA_DeleteTiles (u8 *screen*, u8 *bg_select*)

Delete a tilest in memory. Note that loading a tileset automatically deletes the preceding one, so you won't need to use this function often.

Parameters:

screen Choose the screen (0 or 1)

bg_select Background number to load (from 0 to 3)

3.5.4.6 void PA_DeleteMap (u8 *screen*, u8 *bg_select*)

Delete a map in memory. Note that loading a map automatically deletes the preceding one, so you won't need to use this function often.

Parameters:

screen Choose the screen (0 or 1)

bg_select Background number to load (from 0 to 3)

3.5.4.7 static inline void PA_DeleteBg (u8 *screen*, u8 *bg_select*) [inline, static]

Delete and reset a complete background.

Parameters:

screen Choose the screen (0 or 1)

bg_select Background number to load (from 0 to 3)

3.5.4.8 void PA_LoadBgMap (u8 *screen*, u8 *bg_select*, void * *bg_map*, u8 *bg_size*)

Load a background's map info.

Parameters:

screen Choose the screen (0 or 1)

bg_select Background number to load (from 0 to 3)

bg_map Name of the map's info (example : (void*)ship_Map) Don't forget the void...

bg_size Background size. This is important, because it also determines whether the Bg is rotatable or not. To use a normal background, use the macros BG_-256X256, BG_256X512, etc...

3.5.4.9 void PA_LoadBackground (u8 *screen*, u8 *bg_number*, const PA_BgStruct * *bg_name*)

Load a background (EasyBg or RotBg).

Parameters:

screen Choose the screen (0 or 1)

bg_number Background number... (0-3)

bg_name Pointer to the background (struct)

Examples:

Backgrounds/Effects/Mode7/source/main.c.

3.5.4.10 static inline void PA_BGScrollX (u8 *screen*, u8 *bg_number*, s32 *x*) [inline, static]

Scroll horizontaly a Tiled background.

Parameters:

screen Choose the screen (0 or 1)

bg_number Background number (0-3)

x X value to scroll

3.5.4.11 static inline void PA_BGScrollY (u8 *screen*, u8 *bg_number*, s32 *y*) [inline, static]

Scroll vertically a Tiled background.

Parameters:

screen Choose the screen (0 or 1)

bg_number Background number (0-3)

y Y value to scroll

3.5.4.12 static inline void PA_SetMapTile (u8 *screen*, u8 *bg_select*, s16 *x*, s16 *y*, s16 *tile_number*) [inline, static]

Change the tile gfx used by a given tile in the map.

Parameters:

screen Choose the screen (0 or 1)

bg_select Background number (0-3)

x X value of the tile to change

y Y value of the map tile to change

tile_number New tile number to put

3.5.4.13 static inline void PA_SetLargeMapTile (u8 *screen*, u8 *bg_select*, s32 *x*, s32 *y*, u32 *tile_info*) [inline, static]

Change the tile info used by a given tile in the map, only for big background (512 large or wide).

Parameters:

screen Choose the screen (0 or 1)

bg_select Background number (0-3)

x X value of the tile to change

y Y value of the map tile to change

tile_info New tile to put (tile + palette + flips...)

3.5.4.14 void PA_SetMapTileHflip (u8 *screen*, u8 *bg_select*, u8 *x*, u8 *y*, u8 *hflip*) [inline, static]

Flip a given tile horizontaly.

Parameters:

screen Choose the screen (0 or 1)

bg_select Background number (0-3)

x X value of the tile to change

y Y value of the map tile to change

hflip Set the map tile to horizontal flip

3.5.4.15 static inline void PA_SetMapTileVflip (u8 *screen*, u8 *bg_select*, u8 *x*, u8 *y*, u8 *vflip*) [inline, static]

Flip a given tile verticaly.

Parameters:

screen Choose the screen (0 or 1)

bg_select Background number (0-3)

x X value of the tile to change

y Y value of the map tile to change

vflip Set the map tile to vertical flip

3.5.4.16 static inline void PA_SetMapTilePal (u8 *screen*, u8 *bg_select*, u8 *x*, u8 *y*, u8 *palette_number*) [inline, static]

Change the 16 color palette used by a tile. Works only if the Bg is in 16 colors...

Parameters:

screen Choose the screen (0 or 1)

bg_select Background number (0-3)

x X value of the tile to change

y Y value of the map tile to change

palette_number Palette number (0-15)

3.5.4.17 static inline void PA_SetBgPrio (u8 *screen*, u8 *bg*, u8 *prio*) [inline, static]

Change a backgrounds priority.

Parameters:

screen Choose the screen (0 or 1)

bg Background...

prio Priority level (0-3, 0 being the highest)

3.5.4.18 static inline void PA_SetBgPrioSeq (u8 *screen*, u8 *priority0*, u8 *priority1*, u8 *priority2*, u8 *priority3*) [inline, static]

Change all the background priorities to a given background order.

Parameters:

screen Choose the screen (0 or 1)

priority0 Background to show on top
priority1 Next one...
priority2 Next one...
priority3 Last one...

3.5.4.19 static inline void PA_ClearBg (u8 *screen*, u8 *bg_select*) [inline, static]

Erase a given background (just the tilemap).

Parameters:

screen Choose de screen (0 or 1)
bg_select Background...

3.5.4.20 void PA_EasyBgScrollX (u8 *screen*, u8 *bg_number*, s32 *x*)

Scroll horizontaly any background.

Parameters:

screen Choose the screen (0 or 1)
bg_number Background number (0-3)
x X value to scroll

3.5.4.21 void PA_EasyBgScrollY (u8 *screen*, u8 *bg_number*, s32 *y*)

Scroll vertically any background.

Parameters:

screen Choose the screen (0 or 1)
bg_number Background number (0-3)
y Y value to scroll

3.5.4.22 static inline void PA_EasyBgScrollXY (u8 *screen*, u8 *bg_number*, s32 *x*, s32 *y*) [inline, static]

Scroll horizontaly and vertically any background.

Parameters:

screen Choose the screen (0 or 1)
bg_number Background number (0-3)
x X value to scroll
y Y value to scroll

3.5.4.23 static inline u8 PA_EasyBgGetPixel (u8 *screen*, u8 *bg_number*, s32 *x*, s32 *y*) [inline, static]

Returns the color (number in the palette) of the screen pixel...

Parameters:

screen Choose the screen (0 or 1)

bg_number Background number (0-3)

x X screen pixel position

y Y screen pixel position

3.5.4.24 static inline u16 PA_EasyBgGetPixelCol (u8 *screen*, u8 *bg_number*, s32 *x*, s32 *y*) [inline, static]

Returns the color (u16 value) of the screen pixel...

Parameters:

screen Choose the screen (0 or 1)

bg_number Background number (0-3)

x X screen pixel position

y Y screen pixel position

3.5.4.25 static inline void PA_SetBgWrap (u8 *screen*, u8 *bg*, u8 *wrap*) [inline, static]

Set on/off the background wrapping (for rotating, 8bit, and 16bit backgrounds).

Parameters:

screen Choose the screen (0 or 1)

bg Background number (0-3)

wrap Wrap around on or off...

Examples:

Backgrounds/Effects/Mode7/source/main.c.

3.5.4.26 static inline void PA_InitParallaxX (u8 *screen*, s32 *bg0*, s32 *bg1*, s32 *bg2*, s32 *bg3*) [inline, static]

Initialise Parallax Scrolling for multiple backgrounds, horizontaly. Chose the speed at which each background will scroll compared to the others. Then use PA_-ParallaxScrollX to scroll...

Parameters:

screen Chose de screen (0 or 1)

bg0 Value for the first background (0). Set to 256 for normal scroll speed, lower for lower speed (128 is half speed...), higher for faster (512 is twice as fast...). You can set negative values. 0 inactivates parallax scrolling for this background

bg1 Same thing for Background 1

bg2 Same thing for Background 2

bg3 Same thing for Background 3

3.5.4.27 static inline void PA_InitParallaxY (u8 *screen*, s32 *bg0*, s32 *bg1*, s32 *bg2*, s32 *bg3*) [inline, static]

Initialise Parallax Scrolling for multiple backgrounds, horizontaly. Chose the speed at which each background will scroll compared to the others. Then use PA_-ParallaxScrollX to scroll...

Parameters:

screen Chose de screen (0 or 1)

bg0 Value for the first background (0). Set to 256 for normal scroll speed, lower for lower speed (128 is half speed...), higher for faster (512 is twice as fast...). You can set negative values. 0 inactivates parallax scrolling for this background

bg1 Same thing for Background 1

bg2 Same thing for Background 2

bg3 Same thing for Background 3

3.5.4.28 static inline void PA_ParallaxScrollX (u8 *screen*, s32 *x*) [inline, static]

Scroll the backgrounds.

Parameters:

screen Chose de screen (0 or 1)

x X value to scroll

3.5.4.29 static inline void PA_ParallaxScrollY (u8 *screen*, s32 *y*) [inline, static]

Scroll the backgrounds.

Parameters:

screen Chose de screen (0 or 1)

y Y value to scroll

3.5.4.30 static inline void PA_ParallaxScrollXY (u8 *screen*, s32 *x*, s32 *y*) [inline, static]

Scroll the backgrounds.

Parameters:

screen Chose de screen (0 or 1)

x X value to scroll

y Y value to scroll

3.6 Background Transition Effects

Functions

- void **PA_InitBgTransEx** (u8 screen, u8 bg)
  Init the BgTransition System.
- static void **PA_InitBgTrans** (u8 screen)
  Init the BgTransition System. USES BG0 !! Place your sprite at a priority of 1 or more if you want them to disappear...
- void **PA_BgTransUpDown** (u8 screen, u16 type, u8 vflip, s16 state)
  Up/Down swipping transition effect.
- void **PA_BgTransLeftRight** (u8 screen, u16 type, u8 hflip, s16 state)
  Left/Right swipping transition effect.
- void **PA_BgTransDiag** (u8 screen, u16 type, u8 hflip, u8 vflip, s16 state)
  Diagonal swipping transition effect.
- void **PA_BgTransCenter** (u8 screen, u16 type, u8 invert, s16 state)
  Center transition effect.

3.6.1 Detailed Description

All the different transition effects...

3.6.2 Function Documentation

3.6.2.1 void PA_InitBgTransEx (u8 *screen*, u8 *bg*)

Init the BgTransition System.

Parameters:

screen Chose de screen (0 or 1)
bg Background (0-3)

3.6.2.2 static inline void PA_InitBgTrans (u8 *screen*) `[inline, static]`

Init the BgTransition System. USES BG0 !! Place your sprite at a priority of 1 or more if you want them to disappear...

Parameters:

screen Chose de screen (0 or 1)

3.6.2.3 void PA_BgTransUpDown (u8 *screen*, u16 *type*, u8 *vflip*, s16 *state*)

Up/Down swipping transition effect.

Parameters:

screen Chose de screen (0 or 1)

type BgTrans type... (0-4). Use macros TRANS_ROUND, TRANS_DIAMOND , TRANS_CROSS, TRANS_LINES, or TRANS_STAR

vflip Vertical flip...

state State, from 0 to TRANS_LENGTH. 0 being visible, TRANS_LENGTH invisible

3.6.2.4 void PA_BgTransLeftRight (u8 *screen*, u16 *type*, u8 *hflip*, s16 *state*)

Left/Right swipping transition effect.

Parameters:

screen Chose de screen (0 or 1)

type BgTrans type... (0-4). Use macros TRANS_ROUND, TRANS_DIAMOND , TRANS_CROSS, TRANS_LINES, or TRANS_STAR

hflip Horizontal flip...

state State, from 0 to TRANS_LENGTH. 0 being visible, TRANS_LENGTH invisible

3.6.2.5 void PA_BgTransDiag (u8 *screen*, u16 *type*, u8 *hflip*, u8 *vflip*, s16 *state*)

Diagonal swipping transition effect.

Parameters:

screen Chose de screen (0 or 1)

type BgTrans type... (0-4). Use macros TRANS_ROUND, TRANS_DIAMOND , TRANS_CROSS, TRANS_LINES, or TRANS_STAR

hflip Horizontal flip...

vflip Vertical flip...

state State, from 0 to TRANS_LENGTH. 0 being visible, TRANS_LENGTH invisible

3.6.2.6 void PA_BgTransCenter (u8 *screen*, u16 *type*, u8 *invert*, s16 *state*)

Center transition effect.

Parameters:

screen Chose de screen (0 or 1)

type BgTrans type... (0-4). Use macros TRANS_ROUND, TRANS_DIAMOND , TRANS_CROSS, TRANS_LINES, or TRANS_STAR

invert Invert in/out

state State, from 0 to TRANS_LENGTH. 0 being visible, TRANS_LENGTH invisible

3.7 Debugging utilities

Defines

- #define **PA_Assert**(c, m) ((c) ? ((void)0) : _PA_Assert(#c, m, __FILE__, __LINE__))

  Shows an error message if the condition is not true.

Functions

- bool **PA_IsEmulator** ()

  Detects if the program is running on an emulator.

- void **PA_iDeaS_DebugOutput** (const char *str)

  Outputs text to the iDeaS debugging console.

- void **PA_iDeaS_DebugPrintf** (const char *str,...)

  Outputs formatted text to the iDeaS debugging console.

- void **PA_iDeaS_Breakpoint** ()

  Triggers a breakpoint on iDeaS.

3.7.1 Detailed Description

Some debugging utilities like emulator detecting and iDeaS debug console printing

3.7.2 Define Documentation

3.7.2.1 #define PA_Assert(c, m) ((c) ? ((void)0) : _PA_Assert(#c, m, __FILE__, __LINE__))

Shows an error message if the condition is not true.

Parameters:

c Condition, like MyVar < 128

m Error message

3.7.3 Function Documentation

3.7.3.1 void PA_iDeaS_DebugOutput (const char * *str*)

Outputs text to the iDeaS debugging console.

Parameters:

str The text to output

3.7.3.2 void PA_iDeaS_DebugPrintf (const char * *str*, ...)

Outputs formatted text to the iDeaS debugging console.

Parameters:

str The text to output

3.8 Bitmap mode

Defines

- #define **PA_Get16bitPixel**(screen, x, y) PA_DrawBg[screen][(x) + ((y) << 8)]

  Get the pixel's color in 16 bit Draw mode...

- #define **PA_SetDrawSize**(screen, draw_size) PA_drawsize[screen] = draw_size;

  Set the size of the pen when drawing.

- #define **PA_Load8bitBitmap**(screen, bitmap) DMA_Copy(bitmap, (void∗)PA_DrawBg[screen], 256∗96, DMA_16NOW)

  Load a bitmap on the screen for an 8 bit drawable background.

- #define **PA_Load16bitBitmap**(screen, bitmap)

  Load a bitmap on the screen for an 16 bit drawable background.

- #define **PA_Clear8bitBg**(screen) dmaFillWords(0, (void∗)PA_DrawBg[screen], 256∗96∗2);

  Clears the screen... for an 8 bit drawable background.

- #define **PA_Clear16bitBg**(screen) dmaFillWords(0, (void∗)PA_DrawBg[screen], 256∗192∗2)

  Clears the screen... for an 16 bit drawable background.

Functions

- void **PA_Init8bitBg** (u8 screen, u8 bg_priority)

  Initialise 8 bit draw mode (palette mode)... Chose the screen and the background priority (0-3). This drawable background will replace Background 3, and must be loaded before all other backgrounds. Takes about 3/8 of the VRAM.

- void **PA_InitBig8bitBg** (u8 screen, u8 bg_priority)

  Same as PA_Init8bitBg, but with an available size of 256x256. Takes up a little more space but allows correct vertical scrolling...

- void **PA_Init16bitBg** (u8 screen, u8 bg_priority)

  Initialise 16 bit draw mode (no palette mode, true colors)... Chose the screen and the background priority (0-3). This drawable background will replace Background 3, and must be loaded before all other backgrounds. Takes about 6/8 of the VRAM, so almost all the space !

- static void **PA_Put8bitPixel** (u8 screen, s16 x, s16 y, u8 color)

  Draw a pixel on screen, on an 8 bit background.

- static void **PA_Put2_8bitPixels** (u8 screen, s16 x, s16 y, u16 colors)

  Draw 2 pixels on screen, on an 8 bit background. These pixels are next to another, and the first pixel must be with a pair X. WAY faster than drawing both pixels separately.

- static void **PA_PutDouble8bitPixels** (u8 screen, s16 x, s16 y, u8 color1, u8 color2)

  Draw 2 pixels on screen, on an 8 bit background. These pixels are next to another, and the first pixel must be with a pair X. WAY faster than drawing both pixels separately.

- static void **PA_Put4_8bitPixels** (u8 screen, s16 x, s16 y, u32 colors)

  Draw 4 pixels on screen, on an 8 bit background. These pixels are next to another, and the first pixel must be with a pair X. Fastest way to draw on the screen...

- static u8 **PA_Get8bitPixel** (u8 screen, u8 x, u8 y)

  Get the pixel's color in 8 bit Draw mode...

- static void **PA_Put16bitPixel** (u8 screen, s16 x, s16 y, u16 color)

  Draw a pixel on screen, on an 16 bit background.

- void **PA_Draw8bitLine** (u8 screen, u16 x1, u16 y1, u16 x2, u16 y2, u8 color)

  Draw a line in Draw mode... for 8 bit drawable background.

- void **PA_Draw16bitLine** (u8 screen, u16 x1, u16 y1, u16 x2, u16 y2, u16 color)

  Draw a line in Draw mode... for 16 bit drawable background.

- void **PA_Draw16bitLineEx** (u8 screen, s16 basex, s16 basey, s16 endx, s16 endy, u16 color, s8 size)

  Draw a thick line in Draw mode... for 16 bit drawable background.

- void **PA_Draw8bitLineEx** (u8 screen, s16 basex, s16 basey, s16 endx, s16 endy, u8 color, s8 size)

  Draw a thick line in Draw mode... for 8 bit drawable background.

- void **PA_Draw16bitRect** (u8 screen, s16 basex, s16 basey, s16 endx, s16 endy, u16 color)

  Draw a rectangle in Draw mode... for 16 bit drawable background.

- void **PA_8bitDraw** (u8 screen, u8 color)

  *For 8 bit background : Nice little function that draws on screen ! All you need to do is chose the color, it'll do the rest. If the **PA** (p. 177) VBL isn't initialised, don't forget to update the stylus position every frame... Juste execute PA_Draw every cycle...*

- void **PA_16bitDraw** (u8 screen, u16 color)

  *For 16 bit : Nice little function that draws on screen ! All you need to do is chose the color, it'll do the rest. If the **PA** (p. 177) VBL isn't initialised, don't forget to update the stylus position every frame... Juste execute PA_Draw every cycle...*

- static void **PA_LoadJpeg** (u8 screen, void ∗jpeg)
  Load a jpeg on a 16 bit background... Don't forget to Init the background !
- void **PA_LoadBmpToBuffer** (u16 ∗Buffer, s16 x, s16 y, void ∗bmp, s16 SWidth)
  Load a BMP in a 16 bit Buffer.
- static void **PA_LoadBmpEx** (u8 screen, s16 x, s16 y, void ∗bmp)
  Load a BMP on a 16 bit background... Don't forget to Init the background !
- static void **PA_LoadBmp** (u8 screen, void ∗bmp)
  Load a BMP on a 16 bit background... Don't forget to Init the background !
- static u16 **PA_GetBmpWidth** (void ∗bmpdata)
  Get a BMP's width in pixels.
- static u16 **PA_GetBmpHeight** (void ∗bmpdata)
  Get a BMP's height in pixels.

3.8.1 Detailed Description

Draw on screen, either a pixel or a line, or anything ! Load a Bitmap, a Jpeg...

3.8.2 Define Documentation

3.8.2.1 #define PA_Get16bitPixel(screen, x, y) PA_DrawBg[screen][(x) + ((y) << 8)]

Get the pixel's color in 16 bit Draw mode...

Parameters:

screen Chose de screen (0 or 1)
x X position. Be carefull, if X is not between 0 and 255, it'll give unwanted results
y Y position. Be carefull, if Y is not between 0 and 191, it'll give unwanted results

3.8.2.2 #define PA_SetDrawSize(screen, draw_size) PA_drawsize[screen] = draw_size;

Set the size of the pen when drawing.

Parameters:

screen Chose de screen (0 or 1)
draw_size Size...

3.8.2.3 #define PA_Load8bitBitmap(screen, bitmap) DMA_Copy(bitmap, (void∗)PA_DrawBg[screen], 256∗96, DMA_16NOW)

Load a bitmap on the screen for an 8 bit drawable background.

Parameters:

screen Chose de screen (0 or 1)

bitmap Bitmap name

3.8.2.4 #define PA_Load16bitBitmap(screen, bitmap)

Value:

```
do{u32 PA_temp; \
    for (PA_temp = 0; PA_temp < 256*192; PA_temp++)\
    PA_DrawBg[screen][PA_temp] = bitmap[PA_temp] | (1 << 15);}while(0)
```

Load a bitmap on the screen for an 16 bit drawable background.

Parameters:

screen Chose de screen (0 or 1)

bitmap Bitmap name

3.8.2.5 #define PA_Clear8bitBg(screen) dmaFillWords(0, (void∗)PA_DrawBg[screen], 256∗96∗2);

Clears the screen... for an 8 bit drawable background.

Parameters:

screen Chose de screen (0 or 1)

3.8.2.6 #define PA_Clear16bitBg(screen) dmaFillWords(0, (void∗)PA_DrawBg[screen], 256∗192∗2)

Clears the screen... for an 16 bit drawable background.

Parameters:

screen Chose de screen (0 or 1)

3.8.3 Function Documentation

3.8.3.1 void PA_Init8bitBg (u8 *screen*, u8 *bg_priority*)

Initialise 8 bit draw mode (palette mode)... Chose the screen and the background priority (0-3). This drawable background will replace Background 3, and must be loaded before all other backgrounds. Takes about 3/8 of the VRAM.

Parameters:

screen Chose de screen (0 or 1)

bg_priority Background priority (0-3) Background priority (0-3)

3.8.3.2 void PA_InitBig8bitBg (u8 *screen*, u8 *bg_priority*)

Same as PA_Init8bitBg, but with an available size of 256x256. Takes up a little more space but allows correct vertical scrolling...

Parameters:

screen Chose de screen (0 or 1)

bg_priority Background priority (0-3) Background priority (0-3)

3.8.3.3 void PA_Init16bitBg (u8 *screen*, u8 *bg_priority*)

Initialise 16 bit draw mode (no palette mode, true colors)... Chose the screen and the background priority (0-3). This drawable background will replace Background 3, and must be loaded before all other backgrounds. Takes about 6/8 of the VRAM, so almost all the space !

Parameters:

screen Chose de screen (0 or 1)

bg_priority Background priority (0-3) Background priority (0-3)

3.8.3.4 static inline void PA_Put8bitPixel (u8 *screen*, s16 *x*, s16 *y*, u8 *color*) [inline, static]

Draw a pixel on screen, on an 8 bit background.

Parameters:

screen Chose de screen (0 or 1)

x X position (0-255)

y Y position (0-191)

color Color in the background palette (0-255)

3.8.3.5 static inline void PA_Put2_8bitPixels (u8 *screen*, s16 *x*, s16 *y*, u16 *colors*) [inline, static]

Draw 2 pixels on screen, on an 8 bit background. These pixels are next to another, and the first pixel must be with a pair X. WAY faster than drawing both pixels separately.

Parameters:

screen Chose de screen (0 or 1)

x X position (0-254), must be PAIR

y Y position (0-191)

colors Colors of the first and second pixels (∗256 for the second)

3.8.3.6 static inline void PA_PutDouble8bitPixels (u8 *screen*, s16 *x*, s16 *y*, u8 *color1*, u8 *color2*) [inline, static]

Draw 2 pixels on screen, on an 8 bit background. These pixels are next to another, and the first pixel must be with a pair X. WAY faster than drawing both pixels separately.

Parameters:

screen Chose de screen (0 or 1)

x X position (0-254), must be PAIR

y Y position (0-191)

color1 Color of the first pixel, in the background palette (0-255)

color2 Color of the second pixel, in the background palette (0-255)

3.8.3.7 static inline void PA_Put4_8bitPixels (u8 *screen*, s16 *x*, s16 *y*, u32 *colors*) [inline, static]

Draw 4 pixels on screen, on an 8 bit background. These pixels are next to another, and the first pixel must be with a pair X. Fastest way to draw on the screen...

Parameters:

screen Chose de screen (0 or 1)

x X position (0-254), must be PAIR

y Y position (0-191)

colors Colors of the 4 pixels

3.8.3.8 static inline u8 PA_Get8bitPixel (u8 *screen*, u8 *x*, u8 *y*) [inline, static]

Get the pixel's color in 8 bit Draw mode...

Parameters:

screen Chose de screen (0 or 1)

x X position. Be carefull, if X is not between 0 and 255, it'll give unwanted results

y Y position. Be carefull, if Y is not between 0 and 191, it'll give unwanted results

3.8.3.9 static inline void PA_Put16bitPixel (u8 *screen*, s16 *x*, s16 *y*, u16 *color*) [inline, static]

Draw a pixel on screen, on an 16 bit background.

Parameters:

screen Chose de screen (0 or 1)

x X position (0-255)

y Y position (0-191)

color 16 bit color, obtained using **PA_RGB(red, green, blue)** (p. 94)

3.8.3.10 void PA_Draw8bitLine (u8 *screen*, u16 *x1*, u16 *y1*, u16 *x2*, u16 *y2*, u8 *color*)

Draw a line in Draw mode... for 8 bit drawable background.

Parameters:

screen Chose de screen (0 or 1)

x1 X position of the first point. Be carefull, if X is not between 0 and 255, it'll give unwanted results

y1 Y position of the first point. Be carefull, if Y is not between 0 and 191, it'll give unwanted results

x2 X position of the second point. Be carefull, if X is not between 0 and 255, it'll give unwanted results

y2 Y position of the second point. Be carefull, if Y is not between 0 and 191, it'll give unwanted results

color Color in the background palette (0-255)

3.8.3.11 void PA_Draw16bitLine (u8 *screen*, u16 *x1*, u16 *y1*, u16 *x2*, u16 *y2*, u16 *color*)

Draw a line in Draw mode... for 16 bit drawable background.

Parameters:

screen Chose de screen (0 or 1)

x1 X position of the first point. Be carefull, if X is not between 0 and 255, it'll give unwanted results

y1 Y position of the first point. Be carefull, if Y is not between 0 and 191, it'll give unwanted results

x2 X position of the second point. Be carefull, if X is not between 0 and 255, it'll give unwanted results

y2 Y position of the second point. Be carefull, if Y is not between 0 and 191, it'll give unwanted results

color 15 bits color. You can use the PA_RGB macro to set the RGB values...

3.8.3.12 void PA_Draw16bitLineEx (u8 *screen*, s16 *basex*, s16 *basey*, s16 *endx*, s16 *endy*, u16 *color*, s8 *size*)

Draw a thick line in Draw mode... for 16 bit drawable background.

Parameters:

screen Chose de screen (0 or 1)

basex X position of the first point. Be carefull, if X is not between 0 and 255, it'll give unwanted results

basey Y position of the first point. Be carefull, if Y is not between 0 and 191, it'll give unwanted results

endx X position of the second point. Be carefull, if X is not between 0 and 255, it'll give unwanted results

endy Y position of the second point. Be carefull, if Y is not between 0 and 191, it'll give unwanted results

color 15 bits color. You can use the PA_RGB macro to set the RGB values...

size Width of the line, in pixels

3.8.3.13 void PA_Draw8bitLineEx (u8 *screen*, s16 *basex*, s16 *basey*, s16 *endx*, s16 *endy*, u8 *color*, s8 *size*)

Draw a thick line in Draw mode... for 8 bit drawable background.

Parameters:

screen Chose de screen (0 or 1)

basex X position of the first point. Be carefull, if X is not between 0 and 255, it'll give unwanted results

basey Y position of the first point. Be carefull, if Y is not between 0 and 191, it'll give unwanted results

endx X position of the second point. Be carefull, if X is not between 0 and 255, it'll give unwanted results

endy Y position of the second point. Be carefull, if Y is not between 0 and 191, it'll give unwanted results

color 15 bits color. You can use the PA_RGB macro to set the RGB values...

size Width of the line, in pixels

3.8.3.14 void PA_Draw16bitRect (u8 *screen*, s16 *basex*, s16 *basey*, s16 *endx*, s16 *endy*, u16 *color*)

Draw a rectangle in Draw mode... for 16 bit drawable background.

Parameters:

screen Chose de screen (0 or 1)

basex X position of the first point. Be carefull, if X is not between 0 and 255, it'll give unwanted results

basey Y position of the first point. Be carefull, if Y is not between 0 and 191, it'll give unwanted results

endx X position of the second point. Be carefull, if X is not between 0 and 255, it'll give unwanted results

endy Y position of the second point. Be carefull, if Y is not between 0 and 191, it'll give unwanted results

color 15 bits color. You can use the PA_RGB macro to set the RGB values...

3.8.3.15 PA_8bitDraw (u8 *screen*, u8 *color*)

For 8 bit background : Nice little function that draws on screen ! All you need to do is chose the color, it'll do the rest. If the **PA** (p. 177) VBL isn't initialised, don't forget to update the stylus position every frame... Juste execute PA_Draw every cycle...

Parameters:

screen Chose de screen (0 or 1)

color Color number in the palette (0-255)

3.8.3.16 PA_16bitDraw (u8 *screen*, u16 *color*)

For 16 bit : Nice little function that draws on screen ! All you need to do is chose the color, it'll do the rest. If the **PA** (p. 177) VBL isn't initialised, don't forget to update the stylus position every frame... Juste execute PA_Draw every cycle...

Parameters:

screen Chose de screen (0 or 1)

color 15 bits color. You can use the PA_RGB macro to set the RGB values...

3.8.3.17 static inline void PA_LoadJpeg (u8 *screen*, void * *jpeg*) [inline, static]

Load a jpeg on a 16 bit background... Don't forget to Init the background !

Parameters:

screen Chose de screen (0 or 1)

jpeg jpeg image...

3.8.3.18 void PA_LoadBmpToBuffer (u16 * *Buffer*, s16 *x*, s16 *y*, void * *bmp*, s16 *SWidth*)

Load a BMP in a 16 bit Buffer.

Parameters:

Buffer Buffer...

x X position of the top left corner

y Y position of the top left corner

bmp BMP image...

SWidth Buffer width to use (256 for screen width...)

3.8.3.19 static inline void PA_LoadBmpEx (u8 *screen*, s16 *x*, s16 *y*, void * *bmp*) [inline, static]

Load a BMP on a 16 bit background... Don't forget to Init the background !

Parameters:

screen Chose de screen (0 or 1)

x X position of the top left corner

y Y position of the top left corner

bmp BMP image...

3.8.3.20 static inline void PA_LoadBmp (u8 *screen*, void * *bmp*) [inline, static]

Load a BMP on a 16 bit background... Don't forget to Init the background !

Parameters:

screen Chose de screen (0 or 1)

bmp BMP image...

3.8.3.21 static inline u16 PA_GetBmpWidth (void * *bmp*) [inline, static]

Get a BMP's width in pixels.

Parameters:

bmp BMP image...

3.8.3.22 static inline u16 PA_GetBmpHeight (void * *bmp*) [inline, static]

Get a BMP's height in pixels.

Parameters:

bmp BMP image...

3.9 Fake 16bit bitmap mode

Defines

- #define **PA_LoadFake16bitBitmap**(screen, bitmap) DMA_Copy(bitmap, (void∗)PA_DrawFake16[screen], 256∗192, DMA_16NOW)

  Load a 16 bit bitmap into a fake 16 bit background.

- #define **PA_ClearFake16bitBg**(screen) dmaFillWords(0, (void∗)PA_DrawFake16[screen], 256∗192∗2)

  Clear a fake 16 bit background.

- #define **PA_PutFake16bitPixel**(screen, x, y, color) PA_DrawFake16[screen][(x) + 256 ∗ (y)] = color

  Plots a pixel into a fake 16 bit background.

- #define **PA_GetFake16bitPixel**(screen, x, y) PA_DrawFake16[screen][(x) + 256 ∗ (y)]

  Gets the color of a specified pixel of a fake 16 bit background.

- #define **PA_DrawFake16bitRect**(screen, x1, y1, x2, y2, color)

  Draws a rectangle on a fake 16 bit background.

- #define **PA_Fake16bitLoadBmpEx**(screen, bmp, x, y) PA_LoadBmpToBuffer(PA_DrawFake16[screen], x, y, bmp, 256)

  Load a BMP on a fake 16 bit background... Don't forget to Init the background !

- #define **PA_Fake16bitLoadBmp**(screen, bmp) PA_Fake16bitLoadBmpEx(screen, bmp, 0, 0)

  Load a BMP on a fake 16 bit background... Don't forget to Init the background !

- #define **PA_Fake16bitLoadGif**(screen, gif) PA_Fake16bitLoadGifXY(screen, gif, 0, 0)

  Load a Gif on a fake 16 bit background... Don't forget to Init the background !

- #define **PA_Fake16bitLoadJpeg**(screen, jpeg) JPEG_DecompressImage((u8∗)jpeg, PA_DrawFake16[screen], 256, 192)

  Load a jpeg on a fake 16 bit background... Don't forget to Init the background !

Functions

- void **PA_InitFake16bitBg** (u8 screen, u8 prio)

  Initialize a fake 16 bit background.

- void **PA_DrawFake16bitLine** (u8 screen, u16 x1, u16 y1, u16 x2, u16 y2, u16 color)

Draws a line on a fake 16 bit background.

3.9.1 Detailed Description

Functions to handle fake 16 bit backgrounds that take up less memory than real ones!

3.9.2 Define Documentation

3.9.2.1 #define PA_LoadFake16bitBitmap(screen, bitmap) DMA_Copy(bitmap, (void∗)PA_DrawFake16[screen], 256∗192, DMA_16NOW)

Load a 16 bit bitmap into a fake 16 bit background.

Parameters:

screen Choose the screen (0 or 1)

bitmap Bitmap name

3.9.2.2 #define PA_ClearFake16bitBg(screen) dmaFillWords(0, (void∗)PA_DrawFake16[screen], 256∗192∗2)

Clear a fake 16 bit background.

Parameters:

screen Choose the screen (0 or 1)

3.9.2.3 #define PA_PutFake16bitPixel(screen, x, y, color) PA_DrawFake16[screen][(x) + 256 ∗ (y)] = color

Plots a pixel into a fake 16 bit background.

Parameters:

screen Choose the screen (0 or 1)

x X position of the point. Be carefull, if X is not between 0 and 255, it'll give unwanted results

y Y position of the point. Be carefull, if Y is not between 0 and 191, it'll give unwanted results

color 15 bits color. You can use the PA_RGB macro to set the RGB values...

3.9.2.4 #define PA_GetFake16bitPixel(screen, x, y) PA_DrawFake16[screen][(x) + 256 * (y)]

Gets the color of a specified pixel of a fake 16 bit background.

Parameters:

screen Choose the screen (0 or 1)

x X position of the point. Be carefull, if X is not between 0 and 255, it'll give unwanted results

y Y position of the point. Be carefull, if Y is not between 0 and 191, it'll give unwanted results

3.9.2.5 #define PA_DrawFake16bitRect(screen, x1, y1, x2, y2, color)

Value:

```
do{\
    PA_DrawFake16bitLine(screen, x1, y1, x2, y1, color);\
    PA_DrawFake16bitLine(screen, x1, y1, x1, y2, color);\
    PA_DrawFake16bitLine(screen, x2, y1, x2, y2, color);\
    PA_DrawFake16bitLine(screen, x1, y2, x2, y2, color);}while(0)
```

Draws a rectangle on a fake 16 bit background.

Parameters:

screen Choose the screen (0 or 1)

x1 X position of the first point. Be carefull, if X is not between 0 and 255, it'll give unwanted results

y1 Y position of the first point. Be carefull, if Y is not between 0 and 191, it'll give unwanted results

x2 X position of the second point. Be carefull, if X is not between 0 and 255, it'll give unwanted results

y2 Y position of the second point. Be carefull, if Y is not between 0 and 191, it'll give unwanted results

color 15 bits color. You can use the PA_RGB macro to set the RGB values...

3.9.2.6 #define PA_Fake16bitLoadBmpEx(screen, bmp, x, y) PA_LoadBmpToBuffer(PA_DrawFake16[screen], x, y, bmp, 256)

Load a BMP on a fake 16 bit background... Don't forget to Init the background !

Parameters:

screen Chose de screen (0 or 1)

x X position of the top left corner

y Y position of the top left corner

bmp BMP image...

3.9.2.7 #define PA_Fake16bitLoadBmp(screen, bmp) PA_-Fake16bitLoadBmpEx(screen, bmp, 0, 0)

Load a BMP on a fake 16 bit background... Don't forget to Init the background !

Parameters:

screen Choose the screen (0 or 1)

bmp BMP image...

3.9.2.8 #define PA_Fake16bitLoadGif(screen, gif) PA_-Fake16bitLoadGifXY(screen, gif, 0, 0)

Load a Gif on a fake 16 bit background... Don't forget to Init the background !

Parameters:

screen Chose de screen (0 or 1)

gif Gif image...

3.9.2.9 #define PA_Fake16bitLoadJpeg(screen, jpeg) JPEG_-DecompressImage((u8∗)jpeg, PA_DrawFake16[screen], 256, 192)

Load a jpeg on a fake 16 bit background... Don't forget to Init the background !

Parameters:

screen Chose de screen (0 or 1)

jpeg jpeg image...

3.9.3 Function Documentation

3.9.3.1 void PA_InitFake16bitBg (u8 *screen*, u8 *prio*)

Initialize a fake 16 bit background.

Parameters:

screen Choose the screen (0 or 1)

prio Background priority (from 0 to 3, being 0 the highest)

3.9.3.2 void PA_DrawFake16bitLine (u8 *screen*, u16 *x1*, u16 *y1*, u16 *x2*, u16 *y2*, u16 *color*)

Draws a line on a fake 16 bit background.

Parameters:

screen Choose the screen (0 or 1)

x1 X position of the first point. Be carefull, if X is not between 0 and 255, it'll give unwanted results

y1 Y position of the first point. Be carefull, if Y is not between 0 and 191, it'll give unwanted results

x2 X position of the second point. Be carefull, if X is not between 0 and 255, it'll give unwanted results

y2 Y position of the second point. Be carefull, if Y is not between 0 and 191, it'll give unwanted results

color 15 bits color. You can use the PA_RGB macro to set the RGB values...

3.10 General Functions

Data Structures

- struct **PA_FifoMsg**

  Represents a message sent through Fifo.

- struct **PA_TransferRegion**

  PAlib transfer region type.

Defines

- #define **FIFO_PALIB** FIFO_SOUND

  PAlib Fifo channel number...

- #define **PA_SendFifoMsg**(msg) fifoSendDatamsg(FIFO_PALIB, sizeof(**PA_FifoMsg**), (u8∗) &msg)

  *Send a **PA_FifoMsg** (p. 186) structure to the other CPU.*

- #define **PA_SendFifoVal**(val) fifoSendValue32(FIFO_PALIB, val)

  Send a 32bit value to the other CPU.

- #define **PA_SendFifoCmd** PA_SendFifoVal

  Send a command value to the other CPU (same as PA_SendFifoVal but for readability).

- #define **PA_GetFifoMsg**(msg, bytes) fifoGetDatamsg(FIFO_PALIB, bytes, (u8∗) &msg)

  *Receive a **PA_FifoMsg** (p. 186) structure from the other CPU.*

- #define **PA_FifoRetWait**() while(!fifoCheckValue32(FIFO_PALIB))

  Wait for the other CPU to send a return value.

- #define **PA_FifoRetVal**() fifoGetValue32(FIFO_PALIB)

  Get the other CPU's return value.

- #define **PA_Transfer** ((volatile **PA_TransferRegion**∗) 0x02FFF100)

  PAlib transfer region (used for the storage of data coming from the ARM7). libnds also does this. As TransferRegion was removed we just skip the first 256 bytes.

- #define **PA_LegacyIPCInit**()

  [DEPRECATED] Initialize the legacy IPC system.

- #define **PA_LidClosed**() _PA_LidDown

  Check if the DS is closed. Returns 0 if open, 1 if closed.

- #define **PA_CloseLidSound**(close_sound)

  Check if the DS is closed. If closed, it pauses the DS, and plays a sound.

- #define **PA_CloseLidSound2**(close_sound, open_sound)

  Check if the DS is closed. If closed, it pauses the DS, and plays a sound. The sound system must be initialized before.

- #define **PA_WaitFor**(something) do{while(!(something)) PA_WaitForVBL();}while(0)

  Wait for a specific thing to happen...

Enumerations

- enum { **PA_MSG_INPUT** = 0x7000, **PA_MSG_MIC** = 0x7100, **PA_MSG_DSLBRIGHT** = 0x7102, **PA_MSG_PSG** = 0x7103 }

  PA_FifoMsg *(*p. *186) message types.*

- enum { **PA_MSG_MICSTOP** = 0x7101 }

  PA_SendFifoCmd() *(*p. *65) commands.*

Functions

- static u32 **PA_FifoGetRetVal** ()

  Inline function to ease the getting of the return value (wait + get).

- void **PA_Init** ()

  Initialise the library. Should be used at the beginning of main().

- void **PA_InitFifo** ()

  Initialize the Fifo system. It is automatically done in ***PA_Init()*** *(*p. *66).*

- void **PA_Init2D** ()

  Resets to 2D state after using 3D functions.

- void **PA_SetVideoMode** (u8 screen, u8 mode)

  Change the video mode... Use this with caution.

- void **PA_UpdateUserInfo** (void)

  Updates the user info. This is automatically done in PA_Init. You can then get any info with the following variables : PA_UserInfo.Color (favorite color), .BdayDay, .BdayMonth, .AlarmHour, .AlarmMinute, .Name, .NameLength, .Message, .MessageLength, .Language.

- void **PA_UpdateRTC** (void)

  *Updates the Real Time Clock, with info on the current date and hour. Automatically updated in the **PA** (p. 177) VBL... Get the info with PA_RTC.Minutes, .Hour, .Seconds, .Day, .Month, and .Year.*

- static void **PA_SwitchScreens** ()

  Switch the bottom and top screens...

- static void **PA_SetAutoCheckLid** (u8 on)

  Automatically check if the DS is closed in PA_WaitForVBL.

- static void **PA_SetLedBlink** (u8 blink, u8 speed)

  Set teh DS Led blinking.

- u8 **PA_CheckLid** ()

  Check if the DS is closed. If closed, it pauses the DS, and returns 1.

- static void **PA_WaitForVBL** ()

  Wait for the VBlank to occur.

- static void **PA_SetScreenLight** (u8 screen, u8 light)

  Set on or off the screen's light.

- static void **PA_SetDSLBrightness** (u8 level)

  Set the DS Lite Light level...

- bool **PA_Locate** (char ∗start, char ∗target, bool isDir, int depth, char ∗result)

  Find a directory in the file system within a given depth.

- void **PA_Error** (const char ∗text)

  Displays an error message.

3.10.1 Detailed Description

Initialise the lib, and other general functions...

3.10.2 Define Documentation

3.10.2.1 #define PA_LegacyIPCInit()

Value:

```
do{ \
    memset((void*) &PA_IPC, 0, sizeof(PA_IPCType)); \
    PA_Transfer->mailData = (u32)(&PA_IPC); \
}while(0)
```

[DEPRECATED] Initialize the legacy IPC system.

Deprecated

3.10.2.2 #define PA_CloseLidSound(close_sound)

Value:

```
do{\
    if(PA_LidClosed()){\
        PA_PlaySimpleSound(close_sound);\
        PA_CheckLid(); \
    }}while(0)
```

Check if the DS is closed. If closed, it pauses the DS, and plays a sound.

Parameters:

close_sound Sound to play, check the sounds doc if you're not sure what to do here

3.10.2.3 #define PA_CloseLidSound2(close_sound, open_sound)

Value:

```
do{\
    if(PA_LidClosed()){\
        PA_PlaySimpleSound(close_sound);\
        PA_CheckLid(); \
        PA_PlaySimpleSound(open_sound); \
    }}while(0)
```

Check if the DS is closed. If closed, it pauses the DS, and plays a sound. The sound system must be initialized before.

Parameters:

close_sound Sound to play when closes, check the sounds doc if you're not sure what to do here

open_sound Sound to play when opens, check the sounds doc if you're not sure what to do here

3.10.2.4 #define PA_WaitFor(something) do{while(!(something)) PA_WaitForVBL();}while(0)

Wait for a specific thing to happen...

Parameters:

something Thing to wait for, like Pad.Newpress.A, or Stylus.Newpress, etc...

3.10.3 Enumeration Type Documentation

3.10.3.1 anonymous enum

PA_FifoMsg (p. 186) message types.

Enumerator:

PA_MSG_INPUT Input message (ARM7->ARM9).

PA_MSG_MIC Microphone record message (ARM9->ARM7).

PA_MSG_DSLBRIGHT DS lite screen brightness message (ARM9->ARM7).

PA_MSG_PSG PSG play message (ARM9->ARM7).

3.10.3.2 anonymous enum

PA_SendFifoCmd() (p. 65) commands.

Enumerator:

PA_MSG_MICSTOP Microphone stop recording message (ARM9->ARM7).

3.10.4 Function Documentation

3.10.4.1 void PA_SetVideoMode (u8 *screen*, u8 *mode*)

Change the video mode... Use this with caution.

Parameters:

screen Screen...

mode Mode 0 for normal, 1 for 1 rotating backgrounds, 2 for 2

Examples:

Backgrounds/Effects/Mode7/source/main.c.

3.10.4.2 static inline void PA_SetAutoCheckLid (u8 *on*) `[inline, static]`

Automatically check if the DS is closed in PA_WaitForVBL.

Parameters:

on 1 for on, 0 for off

3.10.4.3 static void PA_SetLedBlink (u8 *blink*, u8 *speed*) [inline, static]

Set teh DS Led blinking.

Parameters:

blink 1 for blinking, 0 for always on

speed Speed : 0 for slow, 1 for fast

3.10.4.4 void PA_SetScreenLight (u8 *screen*, u8 *light*) [inline, static]

Set on or off the screen's light.

Parameters:

screen Screen...

light Light, 1 for on, 0 for off

3.10.4.5 static inline void PA_SetDSLBrightness (u8 *level*) [inline, static]

Set the DS Lite Light level...

Parameters:

level Light level (0-3)

3.10.4.6 bool PA_Locate (char * *start*, char * *target*, bool *isDir*, int *depth*, char * *result*)

Find a directory in the file system within a given depth.

Parameters:

start from which directory to start, use "/" to search from the root

target what to look for: the name of a file or directory

isDir look for a directory or a file?

depth how much depth level (in number of directories) to traverse; limiting this speeds up the search on crowded cards. A reasonable value is, for example, 3.

result pointer to a buffer where the result will be stored

Returns:

true if the target was found

3.11 Gif functions

Functions

- static u16 **PA_GetGifWidth** (void *gif)

  Get a Gif's width in pixels.

- static u16 **PA_GetGifHeight** (void *gif)

  Get a Gif's height in pixels.

- static void **PA_LoadGifXY** (u8 screen, s16 x, s16 y, void *gif)

  Load a Gif on a 16 bit background... Don't forget to Init the background !

- static void **PA_LoadGif** (u8 screen, void *gif)

  Load a Gif on a 16 bit background... Don't forget to Init the background !

- static void **PA_GifAnimSpeed** (float speed)

  Set the gif's speed.

- static void **PA_GifAnimStop** (void)

  Stop a Gif animation.

- static void **PA_GifAnimPause** (void)

  Pause a Gif animation.

- static void **PA_GifSetStartFrame** (s32 StartFrame)

  Set the Gif's starting frame number.

- static void **PA_GifSetEndFrame** (s32 EndFrame)

  Set the Gif's ending frame number.

- static s32 **PA_GifGetFrame** (void)

  Return's the gif's current frame.

3.11.1 Detailed Description

Manages everything about gif files.

3.11.2 Function Documentation

3.11.2.1 static inline u16 PA_GetGifWidth (void * *gif*) [inline, static]

Get a Gif's width in pixels.

Parameters:

gif Gif image...

3.11.2.2 static inline u16 PA_GetGifHeight (void * *gif*) [inline, static]

Get a Gif's height in pixels.

Parameters:

gif Gif image...

3.11.2.3 static inline void PA_LoadGifXY (u8 *screen*, s16 *x*, s16 *y*, void * *gif*) [inline, static]

Load a Gif on a 16 bit background... Don't forget to Init the background !

Parameters:

screen Chose de screen (0 or 1)

x X position on the screen

y Y position on the screen

gif Gif image...

3.11.2.4 static inline void PA_LoadGif (u8 *screen*, void * *gif*) [inline, static]

Load a Gif on a 16 bit background... Don't forget to Init the background !

Parameters:

screen Chose de screen (0 or 1)

gif Gif image...

3.11.2.5 static inline void PA_GifAnimSpeed (float *speed*) [inline, static]

Set the gif's speed.

Parameters:

speed 1 for normal, 2 for 2x, 0.5 for half speed...

3.11.2.6 static inline void PA_GifAnimStop (void) [inline, static]

Stop a Gif animation. Unpause a Gif animation.

3.11.2.7 static inline void PA_GifSetStartFrame (s32 *StartFrame*) [inline, static]

Set the Gif's starting frame number.

Parameters:

StartFrame Starting frame... (0 to start from beginning)

3.11.2.8 static inline void PA_GifSetEndFrame (s32 *EndFrame*) [inline, static]

Set the Gif's ending frame number.

Parameters:

EndFrame Ending frame... (100000 if you want to be sure ^^)

3.12 Keyboard

Defines

- #define **PA_InitKeyboard** PA_LoadDefaultKeyboard

  *Old name for **PA_LoadDefaultKeyboard()*** (p. 75).

- #define **PA_InitCustomKeyboard**(bg_number, keyb_custom)

  [DEPRECATED] Initialise a custom Keyboard on a given background.

- #define **PA_EraseLastKey**() PA_SetLetterPal(PA_Keyboard_Struct.oldX, PA_Keyboard_Struct.oldY, 15)

  Erase the last key lit up (if it didn't on it's own).

Functions

- void **PA_LoadDefaultKeyboard** (u8 bg_number)

  Initialise the default Keyboard on a given background. Uses 16 color palettes 14 and 15 (doesn't mix with text though, don't worry).

- void **PA_LoadKeyboard** (u8 bg_number, const **PA_BgStruct** *keyboard)

  Load a custom Keyboard on a given background.

- char **PA_CheckKeyboard** (void)

  Checks if the keyboard is used, and return the letter :) Use this every turn (even if the stylus isn't pressed).

- static void **PA_ScrollKeyboardX** (s16 x)

  Set the Keyboard's X position.

- static void **PA_ScrollKeyboardY** (s16 y)

  Set the Keyboard's Y position.

- static void **PA_ScrollKeyboardXY** (s16 x, s16 y)

  Set the Keyboard's position.

- static void **PA_KeyboardIn** (s16 x, s16 y)

  Make the keyboard enter to position (x, y), scrolling from the bottom of the screen.

- static void **PA_KeyboardOut** (void)

  Make the keyboard scroll out.

- void **PA_ReloadKeyboardCol** (void)

  Reloads the keyboard's palette, usefull if you changed the background palette.

- static void **PA_SetKeyboardColor** (u8 color1, u8 color2)

  You can change the color used by the keyboard...

- static void **PA_SetKeyboardScreen** (u8 screen)

  Set Keyboard screen. Must be used BEFORE the keyboard init..

3.12.1 Detailed Description

Load a keyboard and have fun

3.12.2 Define Documentation

3.12.2.1 #define PA_InitCustomKeyboard(bg_number, keyb_custom)

Value:

```
do{\
    PA_DEPRECATED_MACRO;\
    PA_LoadBgPal(keyb_screen, bg_number, (void*)keyb_custom##_Pal);\
    PA_LoadSimpleBg(keyb_screen, bg_number, keyb_custom##_Tiles, keyb_custom##_Ma
      p, BG_256X512, 1, 1);\
    PA_Keyboard_Struct.Bg = bg_number;  PA_Keyboard_Struct.Type = 0;    PA_Keyboa
      rd_Struct.Repeat = 0;\
    PA_Keyboard_Struct.Custom = 1;\
    PA_BgInfo[keyb_screen][PA_Keyboard_Struct.Bg].Map = (u32)keyb_custom##_Map;\
}while(0)
```

[DEPRECATED] Initialise a custom Keyboard on a given background.

Deprecated

Parameters:

bg_number Background number (0-3)

keyb_custom Custom Keyboard name, converted as EasyBg

3.12.3 Function Documentation

3.12.3.1 void PA_LoadDefaultKeyboard (u8 *bg_number*)

Initialise the default Keyboard on a given background. Uses 16 color palettes 14 and 15 (doesn't mix with text though, don't worry).

Parameters:

bg_number Background number (0-3)

3.12.3.2 void PA_LoadKeyboard (u8 *bg_number*, const PA_BgStruct * *keyboard*)

Load a custom Keyboard on a given background.

Parameters:

bg_number Background number (0-3)

keyboard Pointer to the keyboard background, converted as EasyBg

3.12.3.3 static inline void PA_ScrollKeyboardX (s16 *x*) [inline, static]

Set the Keyboard's X position.

Parameters:

x X position...

3.12.3.4 static inline void PA_ScrollKeyboardY (s16 *y*) [inline, static]

Set the Keyboard's Y position.

Parameters:

y Y position...

3.12.3.5 static inline void PA_ScrollKeyboardXY (s16 *x*, s16 *y*) [inline, static]

Set the Keyboard's position.

Parameters:

x X position...

y Y position...

3.12.3.6 static inline void PA_KeyboardIn (s16 *x*, s16 *y*) [inline, static]

Make the keyboard enter to position (x, y), scrolling from the bottom of the screen.

Parameters:

x X position...

y Y position...

3.12.3.7 static inline void PA_SetKeyboardColor (u8 *color1*, u8 *color2*) [inline, static]

You can change the color used by the keyboard...

Parameters:

color1 Normal color, 0 for blue, 1 for red, 2 for green

color2 Pressed key color, 0 for blue, 1 for red, 2 for green

3.12.3.8 static inline void PA_SetKeyboardScreen (u8 *screen*) [inline, static]

Set Keyboard screen. Must be used BEFORE the keyboard init..

Parameters:

screen 0 (bottom) or 1 (top)

3.13 Key input system

Defines

- #define **PA_MoveSprite**(sprite) PA_MoveSpriteEx(PA_Screen, sprite, PA_-GetSpriteLx(0, sprite), PA_GetSpriteLy(0, sprite))

  Move a sprite according to the stylus's position. The sprite will be 'hooked' if the stylus passes over it, and then they'll be linked together. Returns 1 if the sprite is moved. You can also get information from PA_MovedSprite.Moving (1 if you are moving a sprite), .Sprite (sprite moved), .X (X position of the center of the sprite), .Y (Y position of the center of the sprite), .Vx (horizontal speed ! useful if you want to make the sprite continue to move when you release the stylus...), and .Vy.

- #define **PA_StylusInZone**(x1, y1, x2, y2) ((Stylus.X>=x1)&&(Stylus.Y>=y1)&&(Stylus.X<x2)&&(Sty

  Check if the stylus is in a given zone... Returns 1 if yes, 0 if not.

Functions

- void **PA_UpdatePad** ()

  Update the Keypad, use it once per frame (in the VBL for example). You can then retrieve the held down keys with Pad.Held.A (or Up, Down...), Newly pressed keys with Pad.Newpress.R, and the just released keys with Pad.Released.Up...

- void **PA_UpdateStylus** ()

  Update the Stylus position. You can then check if the stylus is current in use (Stylus.Held), newly pressed (Stylus.Newpress), or released (Stylus.Released), and get it's position (Stylus.X, Stylus.Y).

- u8 **PA_MoveSpritePix** (u8 sprite)

  Move a sprite according to the stylus's position, only if you touch a sprite's pixel. This is similar to PA_MoveSprite, but slightly slower and requires PA_-InitSpriteDraw(screen, sprite) before. The sprite will be 'hooked' if the stylus passes over it, and then they'll be linked together. Returns 1 if the sprite is moved. You can also get information from PA_MovedSprite.Moving (1 if you are moving a sprite), .Sprite (sprite moved), .X (X position of the top left corner of the sprite), .Y (Y position of the top left corner of the sprite), .Vx (horizontal speed ! useful if you want to make the sprite continue to move when you release the stylus...), and .Vy.

- u8 **PA_MoveSpriteEx** (u8 screen, u8 sprite, u8 lx, u8 ly)

  Move a sprite according to the stylus's position. See PA_MoveSprite for more details... The difference is that here you chose the sprite dimension (lx and ly), which is useful if the sprite is smaller than the DS standard sizes... (for example 20x20...). This will also limit the 'hooking' distance.

- static u8 **PA_MoveSpriteDistance** (u8 sprite, u8 distance)

  Move a sprite according to the stylus's position. See PA_MoveSprite for more details... The difference is that here you chose the hooking distance in pixels.

- static u8 **PA_SpriteStylusOverEx** (u8 sprite, u8 lx, u8 ly)

  Check if the stylus position is over a given sprite (stylus pressed or not).

- static u8 **PA_SpriteTouchedEx** (u8 sprite, u8 lx, u8 ly)

  Check if a given sprite is touched. Returns 1 if touched... You can chose the width and height around the sprite.

- static u8 **PA_SpriteTouched** (u8 sprite)

  Check if a given sprite is touched. Returns 1 if touched...

- static u8 **PA_SpriteStylusOver** (u8 sprite)

  Check if the stylus position is over a given sprite (stylus pressed or not).

3.13.1 Detailed Description

Check which keys are pressed...

3.13.2 Define Documentation

3.13.2.1 #define PA_MoveSprite(sprite) PA_MoveSpriteEx(PA_Screen, sprite, PA_GetSpriteLx(0, sprite), PA_GetSpriteLy(0, sprite))

Move a sprite according to the stylus's position. The sprite will be 'hooked' if the stylus passes over it, and then they'll be linked together. Returns 1 if the sprite is moved. You can also get information from PA_MovedSprite.Moving (1 if you are moving a sprite), .Sprite (sprite moved), .X (X position of the center of the sprite), .Y (Y position of the center of the sprite), .Vx (horizontal speed ! useful if you want to make the sprite continue to move when you release the stylus...), and .Vy.

Parameters:

sprite Object number in the sprite system

3.13.2.2 #define PA_StylusInZone(x1, y1, x2, y2) ((Stylus.X>=x1)&&(Stylus.Y>=y1)&&(Stylus.X<x2)&&(Stylus.Y<y2))

Check if the stylus is in a given zone... Returns 1 if yes, 0 if not.

Parameters:

x1 X value of the upper left corner

y1 Y value of the upper left corner

x2 X value of the lower right corner

y2 Y value of the lower right corner

3.13.3 Function Documentation

3.13.3.1 u8 PA_MoveSpritePix (u8 *sprite*)

Move a sprite according to the stylus's position, only if you touch a sprite's pixel. This is similar to PA_MoveSprite, but slightly slower and requires PA_InitSpriteDraw(screen, sprite) before. The sprite will be 'hooked' if the stylus passes over it, and then they'll be linked together. Returns 1 if the sprite is moved. You can also get information from PA_MovedSprite.Moving (1 if you are moving a sprite), .Sprite (sprite moved), .X (X position of the top left corner of the sprite), .Y (Y position of the top left corner of the sprite), .Vx (horizontal speed ! useful if you want to make the sprite continue to move when you release the stylus...), and .Vy.

Parameters:

sprite Object number in the sprite system

3.13.3.2 u8 PA_MoveSpriteEx (u8 *screen*, u8 *sprite*, u8 *lx*, u8 *ly*)

Move a sprite according to the stylus's position. See PA_MoveSprite for more details... The difference is that here you chose the sprite dimension (lx and ly), which is useful if the sprite is smaller than the DS standard sizes... (for example 20x20...). This will also limit the 'hooking' distance.

Parameters:

screen On what screen to do it

sprite Object number in the sprite system

lx Sprite length

ly Sprite height

3.13.3.3 u8 PA_MoveSpriteDistance (u8 *sprite*, u8 *distance*) [inline, static]

Move a sprite according to the stylus's position. See PA_MoveSprite for more details... The difference is that here you chose the hooking distance in pixels.

Parameters:

sprite Object number in the sprite system

distance Hooking distance

3.13.3.4 static inline u8 PA_SpriteStylusOverEx (u8 *sprite*, u8 *lx*, u8 *ly*) [inline, static]

Check if the stylus position is over a given sprite (stylus pressed or not).

Parameters:

sprite Sprite number in the sprite system

lx Wideness

ly Height

3.13.3.5 static inline u8 PA_SpriteTouchedEx (u8 *sprite*, u8 *lx*, u8 *ly*) [inline, static]

Check if a given sprite is touched. Returns 1 if touched... You can chose the width and height around the sprite.

Parameters:

sprite Sprite number in the sprite system

lx Wideness

ly Height

3.13.3.6 static inline u8 PA_SpriteTouched (u8 *sprite*) [inline, static]

Check if a given sprite is touched. Returns 1 if touched...

Parameters:

sprite Sprite number in the sprite system

3.13.3.7 static inline u8 PA_SpriteStylusOver (u8 *sprite*) [inline, static]

Check if the stylus position is over a given sprite (stylus pressed or not).

Parameters:

sprite Sprite number in the sprite system

3.14 Special controllers

Functions

- bool **PA_DetectGHPad** ()

  Check to see if there's a Guitar Hero pad inserted in slot-2. Returns 1 if there is or 0 if there isn't.

- bool **PA_InitGHPad** ()

  Set up the Guitar Hero pad for use. Returns a 1 if initialization was successful, or a 0 if it wasn't.

- void **PA_DeInitGHPad** ()

  De-initialize the Guitar Hero pad. It's recommended to call this when you won't be using the GH pad anymore.

- void **PA_UpdateGHPad** ()

  Update the values of GHPad. But NOTE: you won't need it if you used PA_InitGHPad as it's done automatically every Vblank.

- bool **PA_DetectPaddle** ()

  Check to see if there's a Taito Paddle inserted in slot-2. Return 1 if there is or 0 if there isn't.

- bool **PA_InitPaddle** ()

  Set up the Taito Paddle for use. Returns a 1 if initialization was successful, or a 0 if it wasn't.

- void **PA_DeInitPaddle** ()

  De-initialize the Taito Paddle. It's recommended to call this when you won't be using the paddle anymore.

- void **PA_UpdatePaddle** ()

  Update the values of Paddle. But NOTE: you won't need it if you used PA_InitPaddle as it's done automatically every Vblank.

3.14.1 Detailed Description

Macros, variables, and prototypes needed for DS controller accessory (Guitar Hero Grip, Taito Paddle, ...) support.

3.15 Math functions

Data Structures

- struct **PA_Point**

  Simple point structure.

Defines

- #define **PA_Cos**(angle) PA_SIN[((angle) + 128)&511]

  Returns the Cos value for an angle. The value goes from -256 to 256... Watch out though : the angle is not in 360 degrees, but in 512 !

- #define **PA_Sin**(angle) PA_SIN[((angle))&511]

  Returns the Sin value for an angle. The value goes from -256 to 256... Watch out though : the angle is not in 360 degrees, but in 256 !

Functions

- static u32 **PA_Rand** ()

  Gives a random number, taken from Ham... This is taken from Ham, I have no credit.

- static void **PA_InitRand** ()

  Auto-seeds the Rand function based on the clock !

- static void **PA_SRand** (s32 r)

  Set the random's seed. This is taken from Ham, I have no credit. I just made it a little shorter/faster (maybe).

- static u32 **PA_RandMax** (u32 max)

  Gives a random number, between 0 and the given number (included).

- static u32 **PA_RandMinMax** (u32 min, u32 max)

  Gives a random number, between the 2 given numbers (included).

- static u64 **PA_Distance** (s32 x1, s32 y1, s32 x2, s32 y2)

  Calculate the distance (squared) between 2 points.

- static u64 **PA_TrueDistance** (s32 x1, s32 y1, s32 x2, s32 y2)

  Calculate the real distance between 2 points. A lot slower than PA_Distance.

- u16 **PA_AdjustAngle** (u16 angle, s16 anglerot, s32 startx, s32 starty, s32 targetx, s32 targety)

  Adjust an angle, for example to calculate in which direction an object shoudl turn.

- static u16 **PA_GetAngle** (s32 startx, s32 starty, s32 targetx, s32 targety)

  Get the angle, from 0 to 511, formed between the horizontal and the line.

- int **PA_mulf32** (int a, int b)

  Multiplies two .12 fixed point integers.

- int **PA_divf32** (int a, int b)

  Divides two .12 fixed point integers.

- int **PA_modf32** (int a, int b)

  Gets the remainder of the division between two .12 fixed point integers (modulo).

- int **PA_sqrtf32** (int a)

  Gets the square root of a .12 fixed point integer.

3.15.1 Detailed Description

Adjust angles, get random values...

3.15.2 Function Documentation

3.15.2.1 void PA_SRand (s32 *r*) [inline, static]

Set the random's seed. This is taken from Ham, I have no credit. I just made it a little shorter/faster (maybe).

Parameters:

r Seed value

3.15.2.2 static inline u32 PA_RandMax (u32 *max*) [inline, static]

Gives a random number, between 0 and the given number (included).

Parameters:

max Maximum included value

3.15.2.3 static inline u32 PA_RandMinMax (u32 *min*, u32 *max*) [inline, static]

Gives a random number, between the 2 given numbers (included).

Parameters:

min Minimum included value

max Maximum included value

3.15.2.4 static inline u32 PA_Distance (s32 *x1*, s32 *y1*, s32 *x2*, s32 *y2*) [inline, static]

Calculate the distance (squared) between 2 points.

Parameters:

x1 X coordinate of the fist point

y1 Y coordinate of the first point

x2 X coordinate of the second point

y2 Y coordinate of the second point

3.15.2.5 static inline u32 PA_TrueDistance (s32 *x1*, s32 *y1*, s32 *x2*, s32 *y2*) [inline, static]

Calculate the real distance between 2 points. A lot slower than PA_Distance.

Parameters:

x1 X coordinate of the fist point

y1 Y coordinate of the first point

x2 X coordinate of the second point

y2 Y coordinate of the second point

3.15.2.6 u16 PA_AdjustAngle (u16 *angle*, s16 *anglerot*, s32 *startx*, s32 *starty*, s32 *targetx*, s32 *targety*)

Adjust an angle, for example to calculate in which direction an object shoudl turn.

Parameters:

angle Base angle, from 0 to 511

anglerot For how much to turn...

startx Initial X position

starty Initial Y position

targetx Target X position

targety Target Y position

3.15.2.7 static inline u16 PA_GetAngle (s32 *startx*, s32 *starty*, s32 *targetx*, s32 *targety*) [inline, static]

Get the angle, from 0 to 511, formed between the horizontal and the line.

Parameters:

startx Initial X position

starty Initial Y position

targetx Target X position

targety Target Y position

3.15.2.8 int PA_mulf32 (int *a*, int *b*)

Multiplies two .12 fixed point integers.

Parameters:

a First number

b Second number

3.15.2.9 int PA_divf32 (int *a*, int *b*)

Divides two .12 fixed point integers.

Parameters:

a First number

b Second number

3.15.2.10 int PA_modf32 (int *a*, int *b*)

Gets the remainder of the division between two .12 fixed point integers (modulo).

Parameters:

a First number

b Second number

3.15.2.11 int PA_sqrtf32 (int *a*)

Gets the square root of a .12 fixed point integer.

Parameters:

a Number

3.16 Microphone

Defines

- #define **PA_MicGetVol**() PA_Transfer->micvol
  Returns the Microphone volume.
- #define **PA_MicStopRecording**() PA_SendFifoCmd(PA_MSG_MICSTOP)
  Stop recording from the microphone.

Functions

- static void **PA_MicStartRecording** (u8 *buffer, u32 length)
  Start recording from the microphone.
- static void **PA_MicReplay** (u8 *buffer, s32 length)
  Play a recorded sound using ASlib.

3.16.1 Detailed Description

Record a sound and replay it...

3.16.2 Function Documentation

3.16.2.1 static inline void PA_MicStartRecording (u8 * *Buffer*, u32 *Length*) [inline, static]

Start recording from the microphone.

Parameters:

Buffer 8bit buffer in which to record the sound

Length Buffer length. To convert seconds to 8bit length you have to multiply the seconds by 16384.

3.16.2.2 static inline void PA_MicReplay (u8 * *Buffer*, s32 *Length*) [inline, static]

Play a recorded sound using ASlib.

Parameters:

Buffer 8bit buffer in which the sound was recorded

Length Buffer length

3.17 Mode 7 commands

Functions

- void **PA_InitMode7** (u8 bg_select)

  Initialize Mode 7 for a given background. You MUST be in video mode 1 or 2.

- static void **PA_DeInitMode7** ()

  DeInitialize Mode 7.

- static void **PA_Mode7Angle** (s16 angle)

  Define the current angle.

- static void **PA_Mode7MoveLeftRight** (s16 x_deplac)

  Move lateraly, so left or right...

- static void **PA_Mode7MoveForwardBack** (s16 z_deplac)

  Move forward or backwards.

- static void **PA_Mode7X** (s16 mode7x)

  Move to a given point on the map.

- static void **PA_Mode7Z** (s16 mode7z)

  Move to a given point on the map.

- static void **PA_Mode7SetPointXZ** (s16 mode7x, s16 mode7z)

  Move to a given point on the map (of coordinates x, z).

- static void **PA_Mode7Height** (s16 mode7y)

  Set the camera height.

3.17.1 Detailed Description

Different commands for Mode 7 :p A big thanks to TONC for these...

3.17.2 Function Documentation

3.17.2.1 void PA_InitMode7 (u8 *bg_select*)

Initialize Mode 7 for a given background. You MUST be in video mode 1 or 2.

Parameters:

bg_select Bg number, 2 in mode 1, 2 or 3 in mode 2

Examples:

Backgrounds/Effects/Mode7/source/main.c.

3.17.2.2 static inline void PA_Mode7Angle (s16 *angle*) [inline, static]

Define the current angle.

Parameters:

angle The angle ranges from 0 to 511...

Examples:

Backgrounds/Effects/Mode7/source/main.c.

3.17.2.3 static inline void PA_Mode7MoveLeftRight (s16 *x_deplac*) [inline, static]

Move lateraly, so left or right...

Parameters:

x_deplac Number of pixels to move left or right

Examples:

Backgrounds/Effects/Mode7/source/main.c.

3.17.2.4 static inline void PA_Mode7MoveForwardBack (s16 *z_deplac*) [inline, static]

Move forward or backwards.

Parameters:

z_deplac Number of pixels to move forward or backwards

Examples:

Backgrounds/Effects/Mode7/source/main.c.

3.17.2.5 static inline void PA_Mode7X (s16 *mode7x*) [inline, static]

Move to a given point on the map.

Parameters:

mode7x X position on the map

3.17.2.6 static inline void PA_Mode7Z (s16 *mode7z*) [inline, static]

Move to a given point on the map.

Parameters:

mode7z Z position on the map

3.17.2.7 static inline void PA_Mode7SetPointXZ (s16 *mode7x*, s16 *mode7z*) [inline, static]

Move to a given point on the map (of coordinates x, z).

Parameters:

mode7x X position on the map

mode7z Z position on the map

3.17.2.8 static inline void PA_Mode7Height (s16 *mode7y*) [inline, static]

Set the camera height.

Parameters:

mode7y Camera Height. By default, 8192. You can set this from 0 to 40 000 (or even more, but then it gets a little small...

Examples:

Backgrounds/Effects/Mode7/source/main.c.

3.18 DS Motion functions

Functions

- static void **PA_MotionInit** (void)

  Turn on the accelerometer.

- static u8 **PA_CheckDSMotion** ()

  Checks whether a DS Motion Card is plugged in.

- static void **PA_MotionToPad** (u8 enable)

  Maps the DS Motion Card to the Pad structure (!!).

Variables

- motion_struct **Motion**

  Motion struct.

3.18.1 Detailed Description

Easy enable and play around with your DS Motion !

3.19 Palette system

Defines

- #define **PA_LoadPal**(palette, source)

  *Load a 256 color palette in the Bg or Sprite palette of screen 0 or 1. Ex : **PA_LoadPal(PALETTE_BG1, bg_pal)** (p. 93);.*

- #define **PA_LoadPal16**(palette, n_palette, source) DMA_Copy((void∗)source, (void∗)(palette + (n_palette << 5)), 16, DMA_16NOW)

  *Load a 16 color palette in the Bg or Sprite palette of screen 0 or 1. Ex : **PA_LoadPal16(PALETTE_BG1, 4, bg_pal)** (p. 94);.*

- #define **PA_LoadSprite16cPal**(screen, n_palette, palette) PA_LoadPal16((PAL_SPRITE0+(0x400∗screen)), (n_palette), palette)

  Load a 16 color palette for sprites.

- #define **PA_RGB**(r, g, b) ((1<<15) | (r) | ((g)<<5) | ((b)<<10))

  Convert Red, Green, and Blue color indexes into a number used in the palette system. Careful : the R, G, B values range from 0 to 31 on gba !

- #define **PA_SetBgPalCol**(screen, color_number, colorRGB) BG_PALETTE[color_number + ((screen) << 9)] = colorRGB

  Change the color of one of the main background palette colors. Not used anymore.

Functions

- static void **PA_Load8bitBgPal** (u8 screen, void ∗Pal)

  Load a palette to be used by the 8bit background.

- void **PA_SetBrightness** (u8 screen, s8 bright)

  Set the screen's brightness.

- static void **PA_SetPalNeg** (u32 palette)

  Set all the palette's color to negative. To undo this, simply negative again...

- static void **PA_SetPal16Neg** (u32 palette, u8 n_palette)

  Set 16 color palette to negative. To undo this, simply negative again...

- void **PA_InitSpriteExtPal** ()

  Initialise 16 palette mode for 256 color sprites. Done by default.

- void **PA_InitBgExtPal** ()

  Initialise 16 palette mode for 256 color backgrounds.

- static void **PA_LoadSpritePal** (u8 screen, u8 palette_number, void ∗palette)

  Load a 256 color palette for Sprites.

- void **PA_LoadBgPalN** (u8 screen, u8 bg_number, u8 pal_number, void ∗palette)

  Load a 256 color palette in the Background palettes, to a given slot.

- static void **PA_LoadBgPal** (u8 screen, u16 bg_number, void ∗palette)

  Load a 256 color palette in the Background palettes.

- void **PA_SetBgPalNCol** (u8 screen, u8 bg_number, u8 pal_number, u8 color_-number, u16 color)

  Change the color of one of the backgrounds' palettes' colors.

- static void **PA_SetBgColor** (u8 screen, u16 color)

  Change the background color of a given screen.

- void **PA_SetSpritePalCol** (u8 screen, u8 pal_number, u8 color_number, u16 color)

  Changes a color in a sprite palette.

- void **PA_3DSetSpritePalCol** (u8 pal_number, u8 color_number, u16 color)

  Changes a color in a 3d sprite palette.

3.19.1 Detailed Description

Load palettes, change palette colors, set the gamma, etc...

3.19.2 Define Documentation

3.19.2.1 #define PA_LoadPal(palette, source)

Value:

```
do{\
    DMA_Copy((void*)source, (void*)palette, 256, DMA_16NOW);\
    if (palette == PAL_SPRITE0) PA_LoadSpritePal(0, 0, (void*)source);\
    if (palette == PAL_SPRITE1) PA_LoadSpritePal(1, 0, (void*)source);\
    if (palette == PAL_BG0) {u8 itemp; for (itemp = 0; itemp < 4; itemp++)
      PA_LoadBgPal(0, itemp, (void*)(source));}\
    if (palette == PAL_BG1) {u8 itemp; for (itemp = 0; itemp < 4; itemp++)
      PA_LoadBgPal(1, itemp, (void*)(source));}}while(0)
```

Load a 256 color palette in the Bg or Sprite palette of screen 0 or 1. Ex : **PA_-LoadPal(PALETTE_BG1, bg_pal)** (p. 93);.

Parameters:

palette Set the Bg palette or Obj palette, screen 0 or 1 : PAL_BG0, PAL_-SPRITE0, PAL_BG1, or PAL_SPRITE1

source Palette name (ex : master_Palette)

3.19.2.2 #define PA_LoadPal16(palette, n_palette, source) DMA_-Copy((void∗)source, (void∗)(palette + (n_palette << 5)), 16, DMA_16NOW)

Load a 16 color palette in the Bg or Sprite palette of screen 0 or 1. Ex : **PA_-LoadPal16(PALETTE_BG1, 4, bg_pal)** (p. 94);.

Parameters:

palette Set the Bg palette or Obj palette, screen 0 or 1 : PAL_BG0, PAL_-SPRITE0, PAL_BG1, or PAL_SPRITE1

n_palette Number of the 16 color palette to load (0-15)

source Palette name (ex : master_Palette)

3.19.2.3 #define PA_LoadSprite16cPal(screen, n_palette, palette) PA_LoadPal16((PAL_SPRITE0+(0x400∗screen)), (n_palette), palette)

Load a 16 color palette for sprites.

Parameters:

screen Screen (0-1)

n_palette Number of the 16 color palette to load (0-15)

palette Palette name (ex : Sprite_Pal)

3.19.2.4 #define PA_RGB(r, g, b) ((1<<15) | (r) | ((g)<<5) | ((b)<<10))

Convert Red, Green, and Blue color indexes into a number used in the palette system. Careful : the R, G, B values range from 0 to 31 on gba !

Parameters:

r Red (0-31)

g Green (0-31)

b Blue (0-31)

3.19.2.5 #define PA_SetBgPalCol(screen, color_number, colorRGB) BG_PALETTE[color_number + ((screen) << 9)] = colorRGB

Change the color of one of the main background palette colors. Not used anymore.

Parameters:

screen Screen...

color_number Color number in palette (0-255)

colorRGB RGB value, like **PA_RGB(31, 31, 31)** (p. 94) for white

3.19.3 Function Documentation

3.19.3.1 static inline void PA_Load8bitBgPal (u8 *screen*, void * *Pal*) [inline, static]

Load a palette to be used by the 8bit background.

Parameters:

screen Screen...

Pal Palette name (ex : master_Palette)

3.19.3.2 void PA_SetBrightness (u8 *screen*, s8 *bright*)

Set the screen's brightness.

Parameters:

screen Chose de screen (0 or 1)

bright Brightness level, from -32 to 32, 0 being neutral

3.19.3.3 static inline void PA_SetPalNeg (u32 *palette*) [inline, static]

Set all the palette's color to negative. To undo this, simply negative again...

Parameters:

palette Set the Bg palette or Obj palette, screen 0 or 1 : PAL_BG0, PAL_SPRITE0, PAL_BG1, or PAL_SPRITE1

3.19.3.4 static inline void PA_SetPal16Neg (u32 *palette*, u8 *n_palette*) [inline, static]

Set 16 color palette to negative. To undo this, simply negative again...

Parameters:

palette Set the Bg palette or Obj palette, screen 0 or 1 : PAL_BG0, PAL_-SPRITE0, PAL_BG1, or PAL_SPRITE1

n_palette Number of the 16 color palette (0-15)

3.19.3.5 void PA_LoadSpritePal (u8 *screen*, u8 *palette_number*, void * *palette*) [inline, static]

Load a 256 color palette for Sprites.

Parameters:

screen Screen...

palette_number Palette number (0-15)

palette Palette to load ((void*)palette_name)

3.19.3.6 void PA_LoadBgPalN (u8 *screen*, u8 *bg_number*, u8 *pal_number*, void * *palette*)

Load a 256 color palette in the Background palettes, to a given slot. Load a 256 color palette in a given Background's palette.

Parameters:

screen Screen...

bg_number Background number (0-3)

pal_number Palette number

palette Palette to load ((void*)palette_name)

screen Screen...

bg_number Background number (0-3)

pal_number Palette number (0-15)

palette Palette to load ((void*)palette_name)

3.19.3.7 void PA_LoadBgPal (u8 *screen*, u16 *bg_number*, void * *palette*) [inline, static]

Load a 256 color palette in the Background palettes.

Parameters:

screen Screen...

bg_number Background number (0-3)

palette Palette to load ((void*)palette_name)

3.19.3.8 void PA_SetBgPalNCol (u8 *screen*, u8 *bg_number*, u8 *pal_number*, u8 *color_number*, u16 *color*)

Change the color of one of the backgrounds' palettes' colors.

Parameters:

screen Screen...

bg_number Background number (0-3)

pal_number Palette number (0-15). Leave to 0 if unsure

color_number Color number in palette (0-255)

color RGB value, like **PA_RGB(31, 31, 31)** (p. 94) for white

3.19.3.9 static inline void PA_SetBgColor (u8 *screen*, u16 *color*) [inline, static]

Change the background color of a given screen.

Parameters:

screen Screen...

color RGB value, like **PA_RGB(31, 31, 31)** (p. 94) for white

3.19.3.10 void PA_SetSpritePalCol (u8 *screen*, u8 *pal_number*, u8 *color_number*, u16 *color*)

Changes a color in a sprite palette.

Parameters:

screen Screen...

pal_number Palette number

color_number Color in the palette

color Color (given by PA_RGB...)

3.19.3.11 void PA_3DSetSpritePalCol (u8 *pal_number*, u8 *color_number*, u16 *color*)

Changes a color in a 3d sprite palette.

Parameters:

pal_number Palette number

color_number Color number in the palette

color Color (given by PA_RGB...)

3.20 Palette system for Dual Screen

Defines

- #define **PA_DualLoadPal**(palette, source)

  Load a 256 color palette in the Bg or Sprite palette of both screens.

- #define **PA_DualLoadPal16**(palette, n_palette, source)

  Load a 16 color palette in the Bg or Sprite palette of both screens.

Functions

- static void **PA_DualSetPalNeg** (u32 palette)

  Set all the palette's color to negative. To undo this, simply negative again...

- static void **PA_DualSetPal16Neg** (u32 palette, u8 n_palette)

  Set 16 color palette to negative. To undo this, simply negative again...

- static void **PA_DualLoadSpritePal** (u8 palette_number, void ∗palette)

  Load a 256 color palette in the Sprite palettes.

- static void **PA_DualLoadBgPal** (u8 bg_number, void ∗palette)

  Load a 256 color palette for a given background.

- static void **PA_DualSetBgColor** (u16 color)

  Change the background color of both screens.

3.20.1 Detailed Description

Load palettes, change palette colors, set the gamma, etc... on both screens !

3.20.2 Define Documentation

3.20.2.1 #define PA_DualLoadPal(palette, source)

Value:

```
do{\
    DMA_Copy((void*)source, (void*)palette, 256, DMA_16NOW);\
    DMA_Copy((void*)(source+1024), (void*)palette, 256, DMA_16NOW);\
    if(palette == PAL_SPRITE)\
        PA_DualLoadSpriteExtPal(0, (void*)palette);\
}while(0)
```

Load a 256 color palette in the Bg or Sprite palette of both screens.

Parameters:

palette Set the Bg palette or Sprite palette : PAL_BG or PAL_SPRITE

source Palette name (ex : master_Palette)

3.20.2.2 #define PA_DualLoadPal16(palette, n_palette, source)

Value:

```
do{\
    DMA_Copy((void*)source, (void*)(palette + (n_palette << 5)), 16, DMA_16NOW);\

    DMA_Copy((void*)source, (void*)(palette + 1024 + (n_palette << 5)), 16, DMA_1
      6NOW);}while(0)
```

Load a 16 color palette in the Bg or Sprite palette of both screens.

Parameters:

palette Set the Bg palette or Obj palette : PAL_BG or PAL_SPRITE

n_palette Number of the 16 color palette to load (0-15)

source Palette name (ex : master_Palette)

3.20.3 Function Documentation

3.20.3.1 static inline void PA_DualSetPalNeg (u32 *palette*) [inline, static]

Set all the palette's color to negative. To undo this, simply negative again...

Parameters:

palette Set the Bg palette or Obj palette : PAL_BG, PAL_SPRITE

3.20.3.2 static inline void PA_DualSetPal16Neg (u32 *palette*, u8 *n_palette*) [inline, static]

Set 16 color palette to negative. To undo this, simply negative again...

Parameters:

palette Set the Bg palette or Obj palette : PAL_BG, PAL_SPRITE

n_palette Number of the 16 color palette (0-15)

3.20.3.3 static inline void PA_DualLoadSpritePal (u8 *palette_number*, void * *palette*) [inline, static]

Load a 256 color palette in the Sprite palettes.

Parameters:

palette_number Palette number (0-15)

palette Palette to load ((void*)palette_name)

3.20.3.4 static inline void PA_DualLoadBgPal (u8 *bg_number*, void * *palette*) [inline, static]

Load a 256 color palette for a given background.

Parameters:

bg_number Background number (0-3)

palette Palette to load ((void*)palette_name)

3.20.3.5 static inline void PA_DualSetBgColor (u16 *color*) [inline, static]

Change the background color of both screens.

Parameters:

color RGB value, like **PA_RGB(31, 31, 31)** (p. 94) for white

3.21 Shape Recognition

Functions

- char **PA_CheckLetter** ()

  Analyzes the drawn shape and returns a letter according to it. 0 if nothing. The drawn shape's string is copied into PA_RecoShape on Stylus Release. You can find a copy of the current letters used here : `http://www.palib.info/Reco/PAGraffiti.gif`.

- static void **PA_RecoAddShape** (char letter, char ∗shape)

  Adds a new shape to the recognition system.

- static void **PA_ResetRecoSys** ()

  Resets the Recognition system.

- static void **PA_UsePAGraffiti** (u8 use)

  *Set on or off the **PA** (p. 177) Graffiti letters. You'll want to turn them off if you plan on using your own shapes....*

3.21.1 Detailed Description

Draw a shape and have it recognized !

3.21.2 Function Documentation

3.21.2.1 static inline void PA_RecoAddShape (char *letter*, char ∗ *shape*) [inline, static]

Adds a new shape to the recognition system.

Parameters:

letter Letter it will return for that shape (you can use any thing, even a number from 1 to 255)

shape 15 characters string given by the recognition system in PA_RecoShape

3.21.2.2 static inline void PA_UsePAGraffiti (u8 *use*) [inline, static]

Set on or off the **PA** (p. 177) Graffiti letters. You'll want to turn them off if you plan on using your own shapes....

Parameters:

use 1/0, on/off...

3.22 Special Effects

Defines

- #define **PA_EnableBgMosaic**(screen, bg) _REG16(REG_BGCNT(screen, bg)) |= (1 << 6)

  Enable the mosaic effect for a given background.

- #define **PA_DisableBgMosaic**(screen, bg) _REG16(REG_BGCNT(screen, bg)) &= ~(1 << 6)

  Disable the mosaic effect for a given background.

- #define **PA_SetBgMosaicXY**(screen, h_size, v_size) do{PA_REG_MOSAIC(screen) &= 255; PA_REG_MOSAIC(screen) |= ((h_size) + ((v_size) << 4));}while(0)

  Set the Mosaic parameters for the backgrounds.

- #define **PA_SetSpriteMosaicXY**(screen, h_size, v_size) do{PA_REG_MOSAIC(screen) &= (255 << 8); PA_REG_MOSAIC(screen) |= (((h_size) << 8) + ((v_size) << 12));}while(0)

  Set the Mosaic parameters for the sprites.

- #define **PA_EnableSpecialFx**(screen, EffectType, FirstTarget, SecondTarget) PA_REG_BLDCNT(screen) = ((FirstTarget) + ((SecondTarget) << 8) + ((EffectType) << 6))

  Enable Special Effects and set whether backgrounds and sprites will use them or not. This also sets the type of Effect.

- #define **PA_DisableSpecialFx**(screen) PA_REG_BLDCNT(screen) = 0

  Disable Special Effects.

- #define **PA_SetSFXAlpha**(screen, Coeff1, Coeff2) PA_REG_BLDALPHA(screen) = (Coeff1) + ((Coeff2) << 8)

  Set the special effect parameters for Alpha-Blending.

3.22.1 Detailed Description

Set the sprite special effects (alpha-blending, luminosity, mosaic effects...)

3.22.2 Define Documentation

3.22.2.1 #define PA_EnableBgMosaic(screen, bg) _REG16(REG_BGCNT(screen, bg)) |= (1 << 6)

Enable the mosaic effect for a given background.

Parameters:

screen Background screen (0 or 1)

bg Background number

3.22.2.2 #define PA_DisableBgMosaic(screen, bg) _-REG16(REG_BGCNT(screen, bg)) &= ∼(1 << 6)

Disable the mosaic effect for a given background.

Parameters:

screen Background screen (0 or 1)

bg Background number

3.22.2.3 #define PA_SetBgMosaicXY(screen, h_size, v_size) do{PA_REG_-MOSAIC(screen) &= 255; PA_REG_MOSAIC(screen) |= ((h_size) + ((v_size) << 4));}while(0)

Set the Mosaic parameters for the backgrounds.

Parameters:

screen Screen...

h_size Horizontal size of the mosaic (1 for 1 pixel, 2 for 2 pixels, etc...)

v_size Vertical size of the mosaic (1 for 1 pixel, 2 for 2 pixels, etc...)

3.22.2.4 #define PA_SetSpriteMosaicXY(screen, h_size, v_size) do{PA_REG_MOSAIC(screen) &= (255 << 8); PA_REG_MOSAIC(screen) |= (((h_size) << 8) + ((v_size) << 12));}while(0)

Set the Mosaic parameters for the sprites.

Parameters:

screen Screen...

h_size Horizontal size of the mosaic (1 for 1 pixel, 2 for 2 pixels, etc...)

v_size Vertical size of the mosaic (1 for 1 pixel, 2 for 2 pixels, etc...)

3.22.2.5 #define PA_EnableSpecialFx(screen, EffectType, FirstTarget, SecondTarget) PA_REG_BLDCNT(screen) = ((FirstTarget) + ((SecondTarget) << 8) + ((EffectType) << 6))

Enable Special Effects and set whether backgrounds and sprites will use them or not. This also sets the type of Effect.

Parameters:

screen Screen...

EffectType Effect Type. 0 for non, 1 for alpha-blending, 2 for brightness increase, and 3 for brightness decrease. You can use the macros SFX_NONE, SFX_-ALPHA, SFX_BRIGHTINC, SFX_BRIGHTDEC

FirstTarget Backgrounds and sprites for which to activate the effect. Use the following macro : SFX_BG0 | SFX_BG1 | SFX_BG2 | SFX_BG3 | SFX_-OBJ | SFX_BD (back drop)

SecondTarget Backgrounds and sprites to be seen behind the alpha-blending. Use the following macro : SFX_BG0 | SFX_BG1 | SFX_BG2 | SFX_BG3 | SFX_OBJ | SFX_BD (back drop)

3.22.2.6 #define PA_DisableSpecialFx(screen) PA_REG_BLDCNT(screen) = 0

Disable Special Effects.

Parameters:

screen Screen...

3.22.2.7 #define PA_SetSFXAlpha(screen, Coeff1, Coeff2) PA_-REG_BLDALPHA(screen) = (Coeff1) + ((Coeff2) << 8)

Set the special effect parameters for Alpha-Blending.

Parameters:

screen Screen...

Coeff1 Coefficient for the first layer, from 0 to 31. Apparently, it's better to set between 0 and 16

Coeff2 Coefficient for the second layer, from 0 to 31. Apparently, it's better to set between 0 and 16

3.23 Sprite system

Defines

- #define **PA_UpdateOAM0**() DMA_Copy((void∗)PA_obj, (void∗)OAM0, 256, DMA_32NOW)

  Update the sprite infos for screen 0 only. Do this in the VBL.

- #define **PA_UpdateOAM1**() DMA_Copy((void∗)PA_obj + 256, (void∗)OAM1, 256, DMA_32NOW)

  Update the sprite infos for screen 1 only. Do this in the VBL.

- #define **PA_UpdateSpriteGfx**(screen, obj_number, obj_data) PA_UpdateGfx(screen, PA_GetSpriteGfx(screen, obj_number), obj_data)

  Update the Gfx of a given sprite.

- #define **PA_SetSpriteRotEnable**(screen, sprite, rotset) do{PA_obj[screen][sprite].atr0 |= OBJ_ROT; PA_obj[screen][sprite].atr1 = (PA_obj[screen][sprite].atr1 & ALL_BUT_ROTSET) + ((rotset) << 9);}while(0)

  Rotate and zoom a sprite.

- #define **PA_SetSpriteRotDisable**(screen, sprite) do{PA_obj[screen][sprite].atr0 &= ALL_BUT(OBJ_ROT); PA_obj[screen][sprite].atr1 &= ALL_BUT_ROTSET;}while(0)

  Stop rotating and zooming a sprite.

- #define **PA_SetSpriteX**(screen, obj, x) PA_obj[screen][obj].atr1 = (PA_obj[screen][obj].atr1 & ALL_BUT(PA_OBJ_X)) + ((x) & PA_OBJ_X)

  Set the X position of a sprite on screen.

- #define **PA_GetSpriteX**(screen, obj) (PA_obj[screen][obj].atr1 & (PA_OBJ_X))

  Get the X position of a sprite on screen.

- #define **PA_SetSpriteY**(screen, obj, y) PA_obj[screen][obj].atr0 = (PA_obj[screen][obj].atr0 & ALL_BUT(PA_OBJ_Y)) + ((y) & PA_OBJ_Y)

  Set the Y position of a sprite on screen.

- #define **PA_GetSpriteY**(screen, obj) (PA_obj[screen][obj].atr0 & PA_OBJ_Y)

  Get the Y position of a sprite on screen.

- #define **PA_SetSpritePal**(screen, obj, pal) PA_obj[screen][obj].atr2 = (PA_obj[screen][obj].atr2 & ALL_BUT_PAL) + ((pal) << 12)

  Set the sprite's palette number.

- #define **PA_GetSpritePal**(screen, obj) (PA_obj[screen][obj].atr2 >> 12)

Get thepalette used by a sprite.

- #define **PA_SetSpriteDblsize**(screen, obj, dblsize) PA_obj[screen][obj].atr0 = (PA_obj[screen][obj].atr0 & ALL_BUT(DBLSIZE)) + ((dblsize) << 9)

  Enable or disable double size for a given sprite.

- #define **PA_GetSpriteDblsize**(screen, obj) ((PA_obj[screen][obj].atr0 & DBLSIZE) >> 9)

  Get the double size state for a given sprite.

- #define **PA_SetSpriteColors**(screen, sprite, n_colors) PA_obj[screen][sprite].atr0 = (PA_obj[screen][sprite].atr0 & ALL_BUT(N_COLORS)) + ((n_colors) << 13)

  Change the sprite's color mode.

- #define **PA_GetSpriteColors**(screen, sprite) ((PA_obj[screen][sprite].atr0 & N_COLORS) >> 13)

  Get a sprite's color mode.

- #define **PA_SetSpriteMode**(screen, sprite, obj_mode) PA_obj[screen][sprite].atr0 = (PA_obj[screen][sprite].atr0 & ALL_BUT(OBJ_MODE)) + ((obj_mode) << 10)

  Set the sprite's mode : 0 for normal, 1 for alpha blending, 2 for window.

- #define **PA_GetSpriteMode**(screen, obj) ((PA_obj[screen][obj].atr0 & OBJ_MODE) >> 10)

  Get the sprite's mode : 0 for normal, 1 for alpha blending, 2 for window.

- #define **PA_SetSpriteMosaic**(screen, obj, mosaic) PA_obj[screen][obj].atr0 = (PA_obj[screen][obj].atr0 & ALL_BUT(OBJ_MOSAIC)) + ((mosaic) << 12)

  Enable or disable mosaic mode for a given sprite.

- #define **PA_GetSpriteMosaic**(screen, obj) ((PA_obj[screen][obj].atr0 & OBJ_MOSAIC) >> 12)

  Get the mosaic mode for a given sprite.

- #define **PA_SetSpriteHflip**(screen, obj, hflip) PA_obj[screen][obj].atr1 = (PA_obj[screen][obj].atr1 & ALL_BUT(OBJ_HFLIP)) + ((hflip) << 12)

  Enable or disable horizontal flip for a given sprite.

- #define **PA_GetSpriteHflip**(screen, obj) ((PA_obj[screen][obj].atr1 & OBJ_HFLIP) >> 12)

  Get the horizontal flip state for a given sprite.

- #define **PA_SetSpriteVflip**(screen, obj, vflip) PA_obj[screen][obj].atr1 = (PA_obj[screen][obj].atr1 & ALL_BUT(OBJ_VFLIP)) + ((vflip) << 13)

  Enable or disable vertical flip for a given sprite.

- #define **PA_GetSpriteVflip**(screen, obj) ((PA_obj[screen][obj].atr1 & OBJ_-VFLIP) >> 13)

  Get the vertical flip state for a given sprite.

- #define **PA_SetSpriteGfx**(screen, obj, gfx) PA_obj[screen][obj].atr2 = (PA_-obj[screen][obj].atr2 & ALL_BUT(OBJ_GFX)) + ((gfx) & OBJ_GFX)

  Change the gfx used by a sprite.

- #define **PA_GetSpriteGfx**(screen, obj) (PA_obj[screen][obj].atr2 & OBJ_-GFX)

  Get the gfx used by a sprite.

- #define **PA_SetSpritePrio**(screen, obj, prio) PA_obj[screen][obj].atr2 = (PA_-obj[screen][obj].atr2 & ALL_BUT(OBJ_PRIO)) + ((prio) << 10)

  Set a sprite's Background priority.

- #define **PA_GetSpritePrio**(screen, obj) ((PA_obj[screen][obj].atr2 & OBJ_-PRIO) >> 10)

  Get a sprite's Background priority.

- #define **PA_GetSpriteLx**(screen, sprite) PA_size[PA_obj[screen][sprite].atr0 >> 14][PA_obj[screen][sprite].atr1 >> 14].lx

  Get a sprite's length.

- #define **PA_GetSpriteLy**(screen, sprite) PA_size[PA_obj[screen][sprite].atr0 >> 14][PA_obj[screen][sprite].atr1 >> 14].ly

  Get a sprite's height.

- #define **PA_CloneSprite**(screen, obj, target) do{PA_obj[screen][obj].atr0 = PA_obj[screen][target].atr0; PA_obj[screen][obj].atr1 = PA_-obj[screen][target].atr1; PA_obj[screen][obj].atr2 = PA_obj[screen][target].atr2; ++obj_per_gfx[screen][PA_GetSpriteGfx(screen, target)];}while(0)

  Clone a sprite. Works only for sprites on the same screen.

Functions

- void **PA_UpdateOAM** (void)

  Update the sprite infos for both screens. Do this in the VBL.

- u16 **PA_CreateGfx** (u8 screen, void *obj_data, u8 obj_shape, u8 obj_size, u8 color_mode)

  Load in mémory a gfx to use later on for a sprite. Returns the gfx's number in memory.

- void **PA_ResetSpriteSys** (void)

Reset the sprite system, memory, etc...

- static void **PA_CreateSprite** (u8 screen, u8 obj_number, void ∗obj_data, u8 obj_shape, u8 obj_size, u8 color_mode, u8 palette, s16 x, s16 y)

  Create a sprite with it's gfx. This is the simple version of the function.

- static void **PA_CreateSpriteEx** (u8 screen, u8 obj_number, void ∗obj_data, u8 obj_shape, u8 obj_size, u8 color_mode, u8 palette, u8 obj_mode, u8 mosaic, u8 hflip, u8 vflip, u8 prio, u8 dblsize, s16 x, s16 y)

  Create a sprite with it's gfx. This is the complex version of the function.

- static void **PA_Create16bitSpriteEx** (u8 screen, u8 obj_number, void ∗obj_data, u8 obj_shape, u8 obj_size, u8 mosaic, u8 hflip, u8 vflip, u8 prio, u8 dblsize, s16 x, s16 y)

  Create a 16 bit sprite with it's gfx. This is the complex version of the function. Warning : a 16bit sprite MUST be 128 pixels large, even if you sprite only takes up a small part on the left...

- static void **PA_Create16bitSpriteFromGfx** (u8 screen, u8 obj_number, u16 gfx, u8 obj_shape, u8 obj_size, s16 x, s16 y)

  Create a 16 bit sprite using a given gfx.

- static void **PA_Create16bitSprite** (u8 screen, u8 obj_number, void ∗obj_data, u8 obj_shape, u8 obj_size, s16 x, s16 y)

  Create a 16 bit sprite with it's gfx. This is the simple version of the function. Warning : a 16bit sprite MUST be 128 pixels large, even if you sprite only takes up a small part on the left...

- static void **PA_CreateSpriteFromGfx** (u8 screen, u8 obj_number, u16 obj_gfx, u8 obj_shape, u8 obj_size, u8 color_mode, u8 palette, s16 x, s16 y)

  Create a sprite with it's gfx. This is the simple version of the function.

- static void **PA_CreateSpriteExFromGfx** (u8 screen, u8 obj_number, u16 obj_gfx, u8 obj_shape, u8 obj_size, u8 color_mode, u8 palette, u8 obj_mode, u8 mosaic, u8 hflip, u8 vflip, u8 prio, u8 dblsize, s16 x, s16 y)

  Create a sprite with it's gfx. This is the complex version of the function.

- static void **PA_UpdateGfx** (u8 screen, u16 gfx_number, void ∗obj_data)

  Update a given Gfx.

- static void **PA_UpdateGfxAndMem** (u8 screen, u8 gfx_number, void ∗obj_data)

  Update the Gfx of a given sprite and updates the PAlib animation pointer... Only for advanced users.

- void **PA_DeleteGfx** (u8 screen, u16 obj_gfx)

  Delete a given Gfx. If a sprite uses this gfx, it'll become invisible.

- void **PA_DeleteSprite** (u8 screen, u8 obj_number)

  Delete a given sprite. If it is the only one to use it's gfx, it'll be deleted too.

- static void **PA_SetRotset** (u8 screen, u8 rotset, s16 angle, u16 zoomx, u16 zoomy)

  Rotate and zoom a sprite.

- static void **PA_SetRotsetNoZoom** (u8 screen, u8 rotset, s16 angle)

  Rotate a sprite without zooming. It's a bit faster than the normal PA_SetRotset function.

- static void **PA_SetRotsetNoAngle** (u8 screen, u8 rotset, u16 zoomx, u16 zoomy)

  Zoom a sprite without rotating. It's a bit faster than the normal PA_SetRotset function.

- static void **PA_SetSpriteXY** (u8 screen, u8 sprite, s16 x, s16 y)

  Set the X and Y position of a sprite on screen.

- static void **PA_Set16bitSpriteAlpha** (u8 screen, u8 sprite, u8 alpha)

  Set the X position of a sprite on screen.

- static void **PA_SetSpriteAnimEx** (u8 screen, u8 sprite, u8 lx, u8 ly, u8 ncolors, s16 animframe)

  Set the animation frame for a given sprite. This function is faster than the normal PA_SetSpriteAnim because it doesn't have to lookup the sprite dimensions...

- static void **PA_SetSpriteAnim** (u8 screen, u8 sprite, s16 animframe)

  Set the animation frame for a given sprite. Same as PA_SetSpriteAnimEx, but a bit slower and easier to use...

- void **PA_StartSpriteAnimEx** (u8 screen, u8 sprite, s16 firstframe, s16 lastframe, s16 speed, u8 type, s16 ncycles)

  Start a sprite animation. Once started, it continues on and on by itself until you stop it !

- static void **PA_StartSpriteAnim** (u8 screen, u8 sprite, s16 firstframe, s16 lastframe, s16 speed)

  Start a sprite animation. Once started, it continues on and on by itself until you stop it !

- static void **PA_StopSpriteAnim** (u8 screen, u8 sprite)

  Stop a sprite animation.

- static void **PA_SetSpriteAnimFrame** (u8 screen, u8 sprite, u16 frame)

  Set the current animation frame number.

- static u16 **PA_GetSpriteAnimFrame** (u8 screen, u8 sprite)

Returns the current animation frame number.

- static void **PA_SetSpriteAnimSpeed** (u8 screen, u8 sprite, s16 speed)

  Set the current animation speed.

- static u16 **PA_GetSpriteAnimSpeed** (u8 screen, u8 sprite)

  Returns the current animation speed.

- static void **PA_SetSpriteNCycles** (u8 screen, u8 sprite, s32 NCycles)

  Set the current animation cycles left (-1 for inifinite loop).

- static s32 **PA_GetSpriteNCycles** (u8 screen, u8 sprite)

  Returns the current number of animation cycles left.

- static void **PA_SpriteAnimPause** (u8 screen, u8 sprite, u8 pause)

  Pause or UnPause a sprite animation.

- static void **PA_SetSpritePixel** (u8 screen, u8 sprite, u8 x, u8 y, u8 color)

  Set a sprite's pixel to a given palette color. Like PA_SetSpritePixelEx, with less options, but a little slower.

- static u8 **PA_GetSpritePixel** (u8 screen, u8 sprite, u8 x, u8 y)

  Get a sprite's pixel color. Like PA_GetSpritePixelEx, with less options, but a little slower.

- static u8 **PA_GetSprite16cPixel** (u8 screen, u8 sprite, u8 x, u8 y)

  Get a 16 color sprite's pixel color.

- void **PA_InitSpriteDraw** (u8 screen, u8 sprite)

  Initialise a sprite to be able to draw on it !

- static void **PA_InitAllSpriteDraw** (void)

  Initialise all the onscreen sprites to draw on them.

- void **PA_InitSpriteExtPrio** (u8 SpritePrio)

  Enable the PAlib sprite priority system. Slower than the normal priority system, but offering 256 levels of priority for the sprites (overrides the sprite number's priority).

3.23.1 Detailed Description

Load Sprite, move them around, rotate them...

3.23.2 Define Documentation

3.23.2.1 #define PA_UpdateSpriteGfx(screen, obj_number, obj_data) PA_UpdateGfx(screen, PA_GetSpriteGfx(screen, obj_number), obj_data)

Update the Gfx of a given sprite.

Parameters:

screen Chose de screen (0 or 1)

obj_number Object number in the sprite system

obj_data Gfx to load

3.23.2.2 #define PA_SetSpriteRotEnable(screen, sprite, rotset) do{PA_obj[screen][sprite].atr0 |= OBJ_ROT; PA_obj[screen][sprite].atr1 = (PA_obj[screen][sprite].atr1 & ALL_BUT_ROTSET) + ((rotset) << 9);}while(0)

Rotate and zoom a sprite.

Parameters:

screen Chose de screen (0 or 1)

sprite Sprite you want to rotate

rotset Rotset you want to give to that sprite (0-31). You can apparently use a rotset for multiple sprites if zoomed/rotated identically...

3.23.2.3 #define PA_SetSpriteRotDisable(screen, sprite) do{PA_obj[screen][sprite].atr0 &= ALL_BUT(OBJ_ROT); PA_obj[screen][sprite].atr1 &= ALL_BUT_ROTSET;}while(0)

Stop rotating and zooming a sprite.

Parameters:

screen Chose de screen (0 or 1)

sprite Sprite you want to rotate

3.23.2.4 #define PA_SetSpriteX(screen, obj, x) PA_obj[screen][obj].atr1 = (PA_obj[screen][obj].atr1 & ALL_BUT(PA_OBJ_X)) + ((x) & PA_OBJ_X)

Set the X position of a sprite on screen.

Parameters:

screen Chose de screen (0 or 1)

obj Object number in the sprite system

x X position

3.23.2.5 #define PA_GetSpriteX(screen, obj) (PA_obj[screen][obj].atr1 & (PA_OBJ_X))

Get the X position of a sprite on screen.

Parameters:

screen Chose de screen (0 or 1)

obj Object number in the sprite system

3.23.2.6 #define PA_SetSpriteY(screen, obj, y) PA_obj[screen][obj].atr0 = (PA_obj[screen][obj].atr0 & ALL_BUT(PA_OBJ_Y)) + ((y) & PA_OBJ_Y)

Set the Y position of a sprite on screen.

Parameters:

screen Chose de screen (0 or 1)

obj Object number in the sprite system

y Y position

3.23.2.7 #define PA_GetSpriteY(screen, obj) (PA_obj[screen][obj].atr0 & PA_OBJ_Y)

Get the Y position of a sprite on screen.

Parameters:

screen Chose de screen (0 or 1)

obj Object number in the sprite system

3.23.2.8 #define PA_SetSpritePal(screen, obj, pal) PA_obj[screen][obj].atr2 = (PA_obj[screen][obj].atr2 & ALL_BUT_PAL) + ((pal) << 12)

Set the sprite's palette number.

Parameters:

screen Chose de screen (0 or 1)

obj Object number in the sprite system

pal Palette number (0 - 15)

3.23.2.9 #define PA_GetSpritePal(screen, obj) (PA_obj[screen][obj].atr2 >> 12)

Get thepalette used by a sprite.

Parameters:

screen Chose de screen (0 or 1)

obj Object number in the sprite system

3.23.2.10 #define PA_SetSpriteDblsize(screen, obj, dblsize) PA_obj[screen][obj].atr0 = (PA_obj[screen][obj].atr0 & ALL_BUT(DBLSIZE)) + ((dblsize) << 9)

Enable or disable double size for a given sprite.

Parameters:

screen Chose de screen (0 or 1)

obj Object number in the sprite system

dblsize 1 to enable doublesize, 0 to disable it...

3.23.2.11 #define PA_GetSpriteDblsize(screen, obj) ((PA_obj[screen][obj].atr0 & DBLSIZE) >> 9)

Get the double size state for a given sprite.

Parameters:

screen Chose de screen (0 or 1)

obj Object number in the sprite system

3.23.2.12 #define PA_SetSpriteColors(screen, sprite, n_colors) PA_obj[screen][sprite].atr0 = (PA_obj[screen][sprite].atr0 & ALL_BUT(N_COLORS)) + ((n_colors) << 13)

Change the sprite's color mode.

Parameters:

screen Chose de screen (0 or 1)

sprite Object number in the sprite system

n_colors 0 for 16 colors, 1 for 256

3.23.2.13 #define PA_GetSpriteColors(screen, sprite) ((PA_obj[screen][sprite].atr0 & N_COLORS) >> 13)

Get a sprite's color mode.

Parameters:

screen Chose de screen (0 or 1)

sprite Object number in the sprite system

3.23.2.14 #define PA_SetSpriteMode(screen, sprite, obj_mode) PA_obj[screen][sprite].atr0 = (PA_obj[screen][sprite].atr0 & ALL_BUT(OBJ_MODE)) + ((obj_mode) << 10)

Set the sprite's mode : 0 for normal, 1 for alpha blending, 2 for window.

Parameters:

screen Chose de screen (0 or 1)

sprite Object number in the sprite system

obj_mode Object mode : 0 for normal, 1 for alpha blending, 2 for window ; not working yet

3.23.2.15 #define PA_GetSpriteMode(screen, obj) ((PA_obj[screen][obj].atr0 & OBJ_MODE) >> 10)

Get the sprite's mode : 0 for normal, 1 for alpha blending, 2 for window.

Parameters:

screen Chose de screen (0 or 1)

obj Object number in the sprite system

3.23.2.16 #define PA_SetSpriteMosaic(screen, obj, mosaic) PA_obj[screen][obj].atr0 = (PA_obj[screen][obj].atr0 & ALL_BUT(OBJ_MOSAIC)) + ((mosaic) << 12)

Enable or disable mosaic mode for a given sprite.

Parameters:

screen Chose de screen (0 or 1)

obj Object number in the sprite system

mosaic Set mosaic on (1) or off (0)

3.23.2.17 #define PA_GetSpriteMosaic(screen, obj) ((PA_obj[screen][obj].atr0 & OBJ_MOSAIC) >> 12)

Get the mosaic mode for a given sprite.

Parameters:

screen Chose de screen (0 or 1)

obj Object number in the sprite system

3.23.2.18 #define PA_SetSpriteHflip(screen, obj, hflip) PA_obj[screen][obj].atr1 = (PA_obj[screen][obj].atr1 & ALL_BUT(OBJ_HFLIP)) + ((hflip) << 12)

Enable or disable horizontal flip for a given sprite.

Parameters:

screen Chose de screen (0 or 1)

obj Object number in the sprite system

hflip Horizontal flip, 1 to enable, 0 to disable...

3.23.2.19 #define PA_GetSpriteHflip(screen, obj) ((PA_obj[screen][obj].atr1 & OBJ_HFLIP) >> 12)

Get the horizontal flip state for a given sprite.

Parameters:

screen Chose de screen (0 or 1)

obj Object number in the sprite system

3.23.2.20 #define PA_SetSpriteVflip(screen, obj, vflip) PA_obj[screen][obj].atr1 = (PA_obj[screen][obj].atr1 & ALL_BUT(OBJ_VFLIP)) + ((vflip) << 13)

Enable or disable vertical flip for a given sprite.

Parameters:

screen Chose de screen (0 or 1)

obj Object number in the sprite system

vflip Vertical flip, 1 to enable, 0 to disable...

3.23.2.21 #define PA_GetSpriteVflip(screen, obj) ((PA_obj[screen][obj].atr1 & OBJ_VFLIP) >> 13)

Get the vertical flip state for a given sprite.

Parameters:

screen Chose de screen (0 or 1)

obj Object number in the sprite system

3.23.2.22 #define PA_SetSpriteGfx(screen, obj, gfx) PA_obj[screen][obj].atr2 = (PA_obj[screen][obj].atr2 & ALL_BUT(OBJ_GFX)) + ((gfx) & OBJ_GFX)

Change the gfx used by a sprite.

Parameters:

screen Chose de screen (0 or 1)

obj Object number in the sprite system

gfx Gfx number ; you can get one by using PA_CreateGfx or **PA_GetSpriteGfx(obj_number)** (p. 116);

3.23.2.23 #define PA_GetSpriteGfx(screen, obj) (PA_obj[screen][obj].atr2 & OBJ_GFX)

Get the gfx used by a sprite.

Parameters:

screen Chose de screen (0 or 1)

obj Object number in the sprite system

3.23.2.24 #define PA_SetSpritePrio(screen, obj, prio) PA_obj[screen][obj].atr2 = (PA_obj[screen][obj].atr2 & ALL_BUT(OBJ_PRIO)) + ((prio) << 10)

Set a sprite's Background priority.

Parameters:

screen Chose de screen (0 or 1)

obj Object number in the sprite system

prio Sprite priority : 0 is over background 0, 1 over Bg 1, etc... (0-3)

3.23.2.25 #define PA_GetSpritePrio(screen, obj) ((PA_obj[screen][obj].atr2 & OBJ_PRIO) >> 10)

Get a sprite's Background priority.

Parameters:

screen Chose de screen (0 or 1)

obj Object number in the sprite system

3.23.2.26 #define PA_GetSpriteLx(screen, sprite) PA_size[PA_obj[screen][sprite].atr0 >> 14][PA_obj[screen][sprite].atr1 >> 14].lx

Get a sprite's length.

Parameters:

screen Chose de screen (0 or 1)

sprite Object number in the sprite system

3.23.2.27 #define PA_GetSpriteLy(screen, sprite) PA_size[PA_obj[screen][sprite].atr0 >> 14][PA_obj[screen][sprite].atr1 >> 14].ly

Get a sprite's height.

Parameters:

screen Chose de screen (0 or 1)

sprite Object number in the sprite system

3.23.2.28 #define PA_CloneSprite(screen, obj, target) do{PA_obj[screen][obj].atr0 = PA_obj[screen][target].atr0; PA_obj[screen][obj].atr1 = PA_obj[screen][target].atr1; PA_obj[screen][obj].atr2 = PA_obj[screen][target].atr2; ++obj_per_gfx[screen][PA_GetSpriteGfx(screen, target)];}while(0)

Clone a sprite. Works only for sprites on the same screen.

Parameters:

screen Chose de screen (0 or 1)

obj Object number in the sprite system

target Target sprite to clone

3.23.3 Function Documentation

3.23.3.1 u16 PA_CreateGfx (u8 *screen*, void * *obj_data*, u8 *obj_shape*, u8 *obj_size*, u8 *color_mode*)

Load in mémory a gfx to use later on for a sprite. Returns the gfx's number in memory.

Parameters:

screen Chose de screen (0 or 1)

obj_data Gfx to load

obj_shape Object shape, from 0 to 2. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

obj_size Object size. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

color_mode 256 or 16 color mode (1 or 0), or 2 for 16bit

3.23.3.2 static inline void PA_CreateSprite (u8 *screen*, u8 *obj_number*, void * *obj_data*, u8 *obj_shape*, u8 *obj_size*, u8 *color_mode*, u8 *palette*, s16 *x*, s16 *y*) `[inline, static]`

Create a sprite with it's gfx. This is the simple version of the function.

Parameters:

screen Chose de screen (0 or 1)

obj_number Object number you want to use (0-127 for each screen seperately).

obj_data Gfx to load

obj_shape Object shape, from 0 to 2. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

obj_size Object size. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

color_mode 256 or 16 color mode (1 or 0).

palette Palette to use (0-15).

x X position of the sprite

y Y position of the sprite

3.23.3.3 static inline void PA_CreateSpriteEx (u8 *screen*, u8 *obj_number*, void * *obj_data*, u8 *obj_shape*, u8 *obj_size*, u8 *color_mode*, u8 *palette*, u8 *obj_mode*, u8 *mosaic*, u8 *hflip*, u8 *vflip*, u8 *prio*, u8 *dblsize*, s16 *x*, s16 *y*) `[inline, static]`

Create a sprite with it's gfx. This is the complex version of the function.

Parameters:

screen Chose de screen (0 or 1)

obj_number Object number you want to use (0-127 for each screen seperately).

obj_data Gfx to load

obj_shape Object shape, from 0 to 2. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

obj_size Object size. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

color_mode 256 or 16 color mode (1 or 0).

palette Palette to use (0-15).

obj_mode Object mode (normal, transparent, window). Not functionnal yet, please leave to 0 for now

mosaic Activate Mosaic for the sprite or not. Not yet functionnal either :p

hflip Horizontal flip on or off...

vflip Vertical flip...

prio Sprite priority regarding backgrounds : in front of which background to show it (0-3)

dblsize Double the possible sprite size. Activate only if you are going to rotate and zoom in the sprite

x X position of the sprite

y Y position of the sprite

3.23.3.4 static inline void PA_Create16bitSpriteEx (u8 *screen*, u8 *obj_number*, void * *obj_data*, u8 *obj_shape*, u8 *obj_size*, u8 *mosaic*, u8 *hflip*, u8 *vflip*, u8 *prio*, u8 *dblsize*, s16 *x*, s16 *y*) [inline, static]

Create a 16 bit sprite with it's gfx. This is the complex version of the function. Warning : a 16bit sprite MUST be 128 pixels large, even if you sprite only takes up a small part on the left...

Parameters:

screen Chose de screen (0 or 1)

obj_number Object number you want to use (0-127 for each screen seperately).

obj_data Gfx to load

obj_shape Object shape, from 0 to 2. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

obj_size Object size. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

mosaic Activate Mosaic for the sprite or not. Not yet functionnal either :p

hflip Horizontal flip on or off...

vflip Vertical flip...

prio Sprite priority regarding backgrounds : in front of which background to show it (0-3)

dblsize Double the possible sprite size. Activate only if you are going to rotate and zoom in the sprite

x X position of the sprite

y Y position of the sprite

3.23.3.5 static inline void PA_Create16bitSpriteFromGfx (u8 *screen*, u8 *obj_number*, u16 *gfx*, u8 *obj_shape*, u8 *obj_size*, s16 *x*, s16 *y*) [inline, static]

Create a 16 bit sprite using a given gfx.

Parameters:

screen Chose de screen (0 or 1)

obj_number Object number you want to use (0-127 for each screen seperately).

gfx Gfx to use

obj_shape Object shape, from 0 to 2. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

obj_size Object size. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

x X position of the sprite

y Y position of the sprite

3.23.3.6 static inline void PA_Create16bitSprite (u8 *screen*, u8 *obj_number*, void * *obj_data*, u8 *obj_shape*, u8 *obj_size*, s16 *x*, s16 *y*) [inline, static]

Create a 16 bit sprite with it's gfx. This is the simple version of the function. Warning : a 16bit sprite MUST be 128 pixels large, even if you sprite only takes up a small part on the left...

Parameters:

screen Chose de screen (0 or 1)

obj_number Object number you want to use (0-127 for each screen seperately).

obj_data Gfx to load

obj_shape Object shape, from 0 to 2. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

obj_size Object size. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

x X position of the sprite

y Y position of the sprite

3.23.3.7 static inline void PA_CreateSpriteFromGfx (u8 *screen*, u8 *obj_number*, u16 *obj_gfx*, u8 *obj_shape*, u8 *obj_size*, u8 *color_mode*, u8 *palette*, s16 *x*, s16 *y*) [inline, static]

Create a sprite with it's gfx. This is the simple version of the function.

Parameters:

screen Chose de screen (0 or 1)

obj_number Object number you want to use (0-127 for each screen seperately).

obj_gfx Memory gfx to use. Get it by using PA_GetSpriteGfx or PA_CreateGfx

obj_shape Object shape, from 0 to 2. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

obj_size Object size. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

color_mode 256 or 16 color mode (1 or 0).

palette Palette to use (0-15).

x X position of the sprite

y Y position of the sprite

3.23.3.8 static inline void PA_CreateSpriteExFromGfx (u8 *screen*, u8 *obj_number*, u16 *obj_gfx*, u8 *obj_shape*, u8 *obj_size*, u8 *color_mode*, u8 *palette*, u8 *obj_mode*, u8 *mosaic*, u8 *hflip*, u8 *vflip*, u8 *prio*, u8 *dblsize*, s16 *x*, s16 *y*) [inline, static]

Create a sprite with it's gfx. This is the complex version of the function.

Parameters:

screen Chose de screen (0 or 1)

obj_number Object number you want to use (0-127 for each screen seperately).

obj_gfx Memory gfx to use. Get it by using PA_GetSpriteGfx or PA_CreateGfx

obj_shape Object shape, from 0 to 2. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

obj_size Object size. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

color_mode 256 or 16 color mode (1 or 0).

palette Palette to use (0-15).

obj_mode Object mode (normal, transparent, window). Not functionnal yet, please leave to 0 for now

mosaic Activate Mosaic for the sprite or not. Not yet functionnal either :p

hflip Horizontal flip on or off...

vflip Vertical flip...

prio Sprite priority regarding backgrounds : in front of which background to show it (0-3)

dblsize Double the possible sprite size. Activate only if you are going to rotate and zoom in the sprite

x X position of the sprite

y Y position of the sprite

3.23.3.9 static inline void PA_UpdateGfx (u8 *screen*, u16 *gfx_number*, void * *obj_data*) [inline, static]

Update a given Gfx.

Parameters:

screen Chose de screen (0 or 1)

gfx_number Gfx number in memory

obj_data Gfx to load

3.23.3.10 static inline void PA_UpdateGfxAndMem (u8 *screen*, u8 *gfx_number*, void * *obj_data*) [inline, static]

Update the Gfx of a given sprite and updates the PAlib animation pointer... Only for advanced users.

Parameters:

screen Chose de screen (0 or 1)

gfx_number Gfx number in memory

obj_data Gfx to load

3.23.3.11 void PA_DeleteGfx (u8 *screen*, u16 *obj_gfx*)

Delete a given Gfx. If a sprite uses this gfx, it'll become invisible.

Parameters:

screen Chose de screen (0 or 1)

obj_gfx Gfx number in memory

3.23.3.12 void PA_DeleteSprite (u8 *screen*, u8 *obj_number*)

Delete a given sprite. If it is the only one to use it's gfx, it'll be deleted too.

Parameters:

screen Chose de screen (0 or 1)

obj_number Sprite number

3.23.3.13 static inline void PA_SetRotset (u8 *screen*, u8 *rotset*, s16 *angle*, u16 *zoomx*, u16 *zoomy*) [inline, static]

Rotate and zoom a sprite.

Parameters:

screen Chose de screen (0 or 1)

rotset Rotset you want to change. To give a sprite a rotset, use PA_SetSpriteRotEnable...

angle Angle, between 0 and 512 (not 360, be carefull)

zoomx Horizontal zoom. 256 is unzoomed, 512 is 2 times smaller, and 128 twice as big... So adjust at will ! :p

zoomy Vertical zoom. 256 is unzoomed, 512 is 2 times smaller, and 128 twice as big... So adjust at will ! :p

3.23.3.14 static inline void PA_SetRotsetNoZoom (u8 *screen*, u8 *rotset*, s16 *angle*) [inline, static]

Rotate a sprite without zooming. It's a bit faster than the normal PA_SetRotset function.

Parameters:

screen Chose de screen (0 or 1)

rotset Rotset you want to change. To give a sprite a rotset, use PA_SetSpriteRotEnable...

angle Angle, between 0 and 512 (not 360, be carefull)

3.23.3.15 static inline void PA_SetRotsetNoAngle (u8 *screen*, u8 *rotset*, u16 *zoomx*, u16 *zoomy*) [inline, static]

Zoom a sprite without rotating. It's a bit faster than the normal PA_SetRotset function.

Parameters:

screen Chose de screen (0 or 1)

rotset Rotset you want to change. To give a sprite a rotset, use PA_SetSpriteRotEnable...

zoomx Horizontal zoom. 256 is unzoomed, 512 is 2 times smaller, and 128 twice as big... So adjust at will ! :p

zoomy Vertical zoom. 256 is unzoomed, 512 is 2 times smaller, and 128 twice as big... So adjust at will ! :p

3.23.3.16 static inline void PA_SetSpriteXY (u8 *screen*, u8 *sprite*, s16 *x*, s16 *y*) [inline, static]

Set the X and Y position of a sprite on screen.

Parameters:

screen Chose de screen (0 or 1)

sprite sprite number in the sprite system

x X position

y X position

3.23.3.17 static inline void PA_Set16bitSpriteAlpha (u8 *screen*, u8 *sprite*, u8 *alpha*) [inline, static]

Set the X position of a sprite on screen.

Parameters:

screen Chose de screen (0 or 1)

sprite Object number in the sprite system, only for 16bit sprites

alpha Alpha parameter, 0-15

3.23.3.18 static inline void PA_SetSpriteAnimEx (u8 *screen*, u8 *sprite*, u8 *lx*, u8 *ly*, u8 *ncolors*, s16 *animframe*) [inline, static]

Set the animation frame for a given sprite. This function is faster than the normal PA_SetSpriteAnim because it doesn't have to lookup the sprite dimensions...

Parameters:

screen Chose de screen (0 or 1)

sprite sprite number in the sprite system

lx Sprite width (8, 16, 32, 64)

ly Sprite height (8, 16, 32, 64)

ncolors Sprite color mode (0 for 16 colors, 1 for 256)

animframe Sprite animation frame (0, 1, 2, etc...)

3.23.3.19 static inline void PA_SetSpriteAnim (u8 *screen*, u8 *sprite*, s16 *animframe*) [inline, static]

Set the animation frame for a given sprite. Same as PA_SetSpriteAnimEx, but a bit slower and easier to use...

Parameters:

screen Chose de screen (0 or 1)

sprite sprite number in the sprite system

animframe Sprite animation frame (0, 1, 2, etc...)

3.23.3.20 void PA_StartSpriteAnimEx (u8 *screen*, u8 *sprite*, s16 *firstframe*, s16 *lastframe*, s16 *speed*, u8 *type*, s16 *ncycles*)

Start a sprite animation. Once started, it continues on and on by itself until you stop it !

Parameters:

screen Chose de screen (0 or 1)

sprite sprite number in the sprite system

firstframe First frame of the animation sequence, most of the time 0...

lastframe Last frame to be displayed. When it gets there, it loops back to the first frame

speed Speed, in frames per second. So speed 1 would mean 1 image per second, so 1 image every game frame

type Defines how you want it to loop. ANIM_LOOP (0) for a normal loop, ANIM_UPDOWN (1) for back and forth animation.

ncycles Number of animation cycles before stopping. If using ANIM_UPDOWN, it takes 2 cycles to come back to the original image

3.23.3.21 static inline void PA_StartSpriteAnim (u8 *screen*, u8 *sprite*, s16 *firstframe*, s16 *lastframe*, s16 *speed*) `[inline, static]`

Start a sprite animation. Once started, it continues on and on by itself until you stop it !

Parameters:

screen Chose de screen (0 or 1)

sprite sprite number in the sprite system

firstframe First frame of the animation sequence, most of the time 0...

lastframe Last frame to be displayed. When it gets there, it loops back to the first frame

speed Speed, in frames per second. So speed 1 would mean 1 image per second, so 1 image every game frame

3.23.3.22 static inline void PA_StopSpriteAnim (u8 *screen*, u8 *sprite*) [inline, static]

Stop a sprite animation.

Parameters:

screen Chose de screen (0 or 1)

sprite sprite number in the sprite system

3.23.3.23 static inline void PA_SetSpriteAnimFrame (u8 *screen*, u8 *sprite*, u16 *frame*) [inline, static]

Set the current animation frame number.

Parameters:

screen Chose de screen (0 or 1)

sprite sprite number in the sprite system

frame Frame number to use...

3.23.3.24 static inline u16 PA_GetSpriteAnimFrame (u8 *screen*, u8 *sprite*) [inline, static]

Returns the current animation frame number.

Parameters:

screen Chose de screen (0 or 1)

sprite sprite number in the sprite system

3.23.3.25 static inline void PA_SetSpriteAnimSpeed (u8 *screen*, u8 *sprite*, s16 *speed*) [inline, static]

Set the current animation speed.

Parameters:

screen Chose de screen (0 or 1)

sprite sprite number in the sprite system

speed Speed, in fps...

3.23.3.26 static inline u16 PA_GetSpriteAnimSpeed (u8 *screen*, u8 *sprite*) [inline, static]

Returns the current animation speed.

Parameters:

screen Chose de screen (0 or 1)

sprite sprite number in the sprite system

3.23.3.27 static inline void PA_SetSpriteNCycles (u8 *screen*, u8 *sprite*, s32 *NCycles*) [inline, static]

Set the current animation cycles left (-1 for inifinite loop).

Parameters:

screen Chose de screen (0 or 1)

sprite sprite number in the sprite system

NCycles Number of cycles

3.23.3.28 static inline s32 PA_GetSpriteNCycles (u8 *screen*, u8 *sprite*) [inline, static]

Returns the current number of animation cycles left.

Parameters:

screen Chose de screen (0 or 1)

sprite sprite number in the sprite system

3.23.3.29 static inline u16 PA_SpriteAnimPause (u8 *screen*, u8 *sprite*, u8 *pause*) [inline, static]

Pause or UnPause a sprite animation.

Parameters:

screen Chose de screen (0 or 1)

sprite sprite number in the sprite system

pause 1 for pause, 0 for unpause

3.23.3.30 static inline void PA_SetSpritePixel (u8 *screen*, u8 *sprite*, u8 *x*, u8 *y*, u8 *color*) [inline, static]

Set a sprite's pixel to a given palette color. Like PA_SetSpritePixelEx, with less options, but a little slower.

Parameters:

screen Chose de screen (0 or 1)

sprite Sprite number in the sprite system

x X coordinate of the pixel to change

y Y coordinate of the pixel to change

color New palette color to put

3.23.3.31 static inline u8 PA_GetSpritePixel (u8 *screen*, u8 *sprite*, u8 *x*, u8 *y*) [inline, static]

Get a sprite's pixel color. Like PA_GetSpritePixelEx, with less options, but a little slower.

Parameters:

screen Chose de screen (0 or 1)

sprite Sprite number in the sprite system

x X coordinate of the pixel

y Y coordinate of the pixel

3.23.3.32 static inline u8 PA_GetSprite16cPixel (u8 *screen*, u8 *sprite*, u8 *x*, u8 *y*) [inline, static]

Get a 16 color sprite's pixel color.

Parameters:

screen Chose de screen (0 or 1)

sprite Sprite number in the sprite system

x X coordinate of the pixel

y Y coordinate of the pixel

3.23.3.33 void PA_InitSpriteDraw (u8 *screen*, u8 *sprite*)

Initialise a sprite to be able to draw on it !

Parameters:

screen Chose de screen (0 or 1)

sprite Sprite number in the sprite system

3.23.3.34 void PA_InitSpriteExtPrio (u8 *SpritePrio*)

Enable the PAlib sprite priority system. Slower than the normal priority system, but offering 256 levels of priority for the sprites (overrides the sprite number's priority).

Parameters:

SpritePrio 1 for on, 0 for off...

3.24 Sprite system for Dual Screen

Functions

- static void **PA_SetScreenSpace** (s16 ScreenSpace)

  Set the space between the 2 screens for the Dual Fonctions. 48 pixels by default.

- static void **PA_DualSetSpriteX** (u8 obj, s16 x)

  Set the X position of a sprite on screen.

- static void **PA_DualSetSpriteY** (u8 obj, s16 y)

  Set the Y position of a sprite on screen.

- static void **PA_DualSetSpriteXY** (u8 sprite, s16 x, s16 y)

  Set the X and Y position of a sprite on screen.

- static void **PA_DualCreateSprite** (u8 obj_number, void ∗obj_data, u8 obj_shape, u8 obj_size, u8 color_mode, u8 palette, s16 x, s16 y)

  Create a sprite with it's gfx, on 2 screens.

- static void **PA_DualCreateSpriteEx** (u8 obj_number, void ∗obj_data, u8 obj_shape, u8 obj_size, u8 color_mode, u8 palette, u8 obj_mode, u8 mosaic, u8 hflip, u8 vflip, u8 prio, u8 dblsize, s16 x, s16 y)

  Create a sprite with it's gfx. This is the complex version of the function.

- static void **PA_DualCreate16bitSpriteEx** (u8 obj_number, void ∗obj_data, u8 obj_shape, u8 obj_size, u8 mosaic, u8 hflip, u8 vflip, u8 prio, u8 dblsize, s16 x, s16 y)

  Create a 16 bit sprite with it's gfx. This is the complex version of the function. Warning : a 16bit sprite MUST be 128 pixels large, even if you sprite only takes up a small part on the left...

- static void **PA_DualCreate16bitSprite** (u8 obj_number, void ∗obj_data, u8 obj_shape, u8 obj_size, s16 x, s16 y)

  Create a 16 bit sprite with it's gfx. This is the simple version of the function. Warning : a 16bit sprite MUST be 128 pixels large, even if you sprite only takes up a small part on the left...

- static void **PA_DualCreateSpriteFromGfx** (u8 obj_number, u16 ∗obj_gfx, u8 obj_shape, u8 obj_size, u8 color_mode, u8 palette, s16 x, s16 y)

  Create a sprite with it's gfx. This is the simple version of the function.

- static void **PA_DualCreateSpriteExFromGfx** (u8 obj_number, u16 ∗obj_gfx, u8 obj_shape, u8 obj_size, u8 color_mode, u8 palette, u8 obj_mode, u8 mosaic, u8 hflip, u8 vflip, u8 prio, u8 dblsize, s16 x, s16 y)

  Create a sprite with it's gfx. This is the complex version of the function.

- static void **PA_DualUpdateSpriteGfx** (u8 obj_number, void ∗obj_data)

  Update the Gfx of a given sprite.

- static void **PA_DualUpdateGfx** (u16 gfx_number, void ∗obj_data)

  Update the Gfx of a given sprite.

- static void **PA_DualDeleteSprite** (u8 obj_number)

  Delete a given sprite. If it is the only one to use it's gfx, it'll be deleted too.

- static void **PA_DualSetSpriteRotEnable** (u8 sprite, u8 rotset)

  Rotate and zoom a sprite.

- static void **PA_DualSetSpriteRotDisable** (u8 sprite)

  Stop rotating and zooming a sprite.

- static void **PA_DualSetRotset** (u8 rotset, s16 angle, u16 zoomx, u16 zoomy)

  Rotate and zoom a sprite.

- static void **PA_DualSetRotsetNoZoom** (u8 rotset, s16 angle)

  Rotate a sprite without zooming. It's a bit faster than the normal PA_SetRotset function.

- static void **PA_DualSetRotsetNoAngle** (u8 rotset, u16 zoomx, u16 zoomy)

  Zoom a sprite without rotating. It's a bit faster than the normal PA_SetRotset function.

- static void **PA_DualSetSpritePal** (u8 obj, u8 pal)

  Set the color palette used by a sprite.

- static void **PA_DualSetSpriteDblsize** (u8 obj, u8 dblsize)

  Enable or disable double size for a given sprite.

- static void **PA_DualSetSpriteColors** (u8 sprite, u8 n_colors)

  Change the sprite's color mode.

- static void **PA_DualSetSpriteMode** (u8 sprite, u8 obj_mode)

  Set the sprite's mode : 0 for normal, 1 for alpha blending, 2 for window.

- static void **PA_DualSetSpriteMosaic** (u8 obj, u8 mosaic)

  Enable or disable mosaic mode for a given sprite.

- static void **PA_DualSetSpriteHflip** (u8 obj, u8 hflip)

  Enable or disable horizontal flip for a given sprite.

- static void **PA_DualSetSpriteVflip** (u8 obj, u8 vflip)

  Enable or disable vertical flip for a given sprite.

- static void **PA_DualSetSpriteGfx** (u8 obj, u16 *gfx)

  Change the gfx used by a sprite.

- static void **PA_DualSetSpritePrio** (u8 obj, u8 prio)

  Set a sprite's Background priority.

- static void **PA_DualCloneSprite** (u8 obj, u8 target)

  Clone a sprite. Works only for sprites on the same screen.

- static void **PA_DualSetSpriteAnimEx** (u8 sprite, u8 lx, u8 ly, u8 ncolors, s16 animframe)

  Set the animation frame for a given sprite. This function is faster than the normal PA_SetSpriteAnim because it doesn't have to lookup the sprite dimensions...

- static void **PA_DualSetSpriteAnim** (u8 sprite, s16 animframe)

  Set the animation frame for a given sprite. Same as PA_SetSpriteAnimEx, but a bit slower and easier to use...

- static void **PA_DualStartSpriteAnimEx** (u8 sprite, s16 firstframe, s16 lastframe, s16 speed, u8 type, s16 ncycles)

  Start a sprite animation for DualSprites. Once started, it continues on and on by itself until you stop it !

- static void **PA_DualStartSpriteAnim** (u8 sprite, s16 firstframe, s16 lastframe, s16 speed)

  Start a sprite animation for DualSprite. Once started, it continues on and on by itself until you stop it !

- static void **PA_DualStopSpriteAnim** (u8 sprite)

  Stop a sprite animation for DualSprites.

- static void **PA_DualSetSpriteAnimFrame** (u8 sprite, u16 frame)

  Set the current animation frame number for DualSprites.

- static u16 **PA_DualGetSpriteAnimFrame** (u8 sprite)

  Returns the current animation frame number for DualSprites.

- static void **PA_DualSetSpriteAnimSpeed** (u8 sprite, s16 speed)

  Set the current animation speed for DualSprites.

- static u16 **PA_DualGetSpriteAnimSpeed** (u8 sprite)

  Returns the current animation speed for DualSprites.

- static void **PA_DualSpriteAnimPause** (u8 sprite, u8 pause)

  Pause or UnPause a sprite animation for DualSprites.

3.24.1 Detailed Description

Load Sprite, move them around, rotate them...

3.24.2 Function Documentation

3.24.2.1 static inline void PA_SetScreenSpace (s16 *ScreenSpace*) [inline, static]

Set the space between the 2 screens for the Dual Fonctions. 48 pixels by default.

Parameters:

ScreenSpace Space in pixels

3.24.2.2 static inline void PA_DualSetSpriteX (u8 *obj*, s16 *x*) [inline, static]

Set the X position of a sprite on screen.

Parameters:

obj Object number in the sprite system

x X position

3.24.2.3 static inline void PA_DualSetSpriteY (u8 *obj*, s16 *y*) [inline, static]

Set the Y position of a sprite on screen.

Parameters:

obj Object number in the sprite system

y Y position

3.24.2.4 static inline void PA_DualSetSpriteXY (u8 *sprite*, s16 *x*, s16 *y*) [inline, static]

Set the X and Y position of a sprite on screen.

Parameters:

sprite sprite number in the sprite system

x X position

y X position

3.24.2.5 static inline void PA_DualCreateSprite (u8 *obj_number*, void * *obj_data*, u8 *obj_shape*, u8 *obj_size*, u8 *color_mode*, u8 *palette*, s16 *x*, s16 *y*) [inline, static]

Create a sprite with it's gfx, on 2 screens.

Parameters:

obj_number Object number you want to use (0-127 for each screen seperately).

obj_data Gfx to load

obj_shape Object shape, from 0 to 2. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

obj_size Object size. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

color_mode 256 or 16 color mode (1 or 0).

palette Palette to use (0-15).

x X position of the sprite

y Y position of the sprite

3.24.2.6 static inline void PA_DualCreateSpriteEx (u8 *obj_number*, void * *obj_data*, u8 *obj_shape*, u8 *obj_size*, u8 *color_mode*, u8 *palette*, u8 *obj_mode*, u8 *mosaic*, u8 *hflip*, u8 *vflip*, u8 *prio*, u8 *dblsize*, s16 *x*, s16 *y*) [inline, static]

Create a sprite with it's gfx. This is the complex version of the function.

Parameters:

obj_number Object number you want to use (0-127 for each screen seperately).

obj_data Gfx to load

obj_shape Object shape, from 0 to 2. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

obj_size Object size. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

color_mode 256 or 16 color mode (1 or 0).

palette Palette to use (0-15).

obj_mode Object mode (normal, transparent, window). Not functionnal yet, please leave to 0 for now

mosaic Activate Mosaic for the sprite or not. Not yet functionnal either :p

hflip Horizontal flip on or off...

vflip Vertical flip...

prio Sprite priority regarding backgrounds : in front of which background to show it (0-3)

dblsize Double the possible sprite size. Activate only if you are going to rotate and zoom in the sprite

x X position of the sprite

y Y position of the sprite

3.24.2.7 static inline void PA_DualCreate16bitSpriteEx (u8 *obj_number*, void * *obj_data*, u8 *obj_shape*, u8 *obj_size*, u8 *mosaic*, u8 *hflip*, u8 *vflip*, u8 *prio*, u8 *dblsize*, s16 *x*, s16 *y*) [inline, static]

Create a 16 bit sprite with it's gfx. This is the complex version of the function. Warning : a 16bit sprite MUST be 128 pixels large, even if you sprite only takes up a small part on the left...

Parameters:

obj_number Object number you want to use (0-127 for each screen seperately).

obj_data Gfx to load

obj_shape Object shape, from 0 to 2. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

obj_size Object size. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

mosaic Activate Mosaic for the sprite or not. Not yet functionnal either :p

hflip Horizontal flip on or off...

vflip Vertical flip...

prio Sprite priority regarding backgrounds : in front of which background to show it (0-3)

dblsize Double the possible sprite size. Activate only if you are going to rotate and zoom in the sprite

x X position of the sprite

y Y position of the sprite

3.24.2.8 static inline void PA_DualCreate16bitSprite (u8 *obj_number*, void * *obj_data*, u8 *obj_shape*, u8 *obj_size*, s16 *x*, s16 *y*) [inline, static]

Create a 16 bit sprite with it's gfx. This is the simple version of the function. Warning : a 16bit sprite MUST be 128 pixels large, even if you sprite only takes up a small part on the left...

Parameters:

obj_number Object number you want to use (0-127 for each screen seperately).

obj_data Gfx to load

obj_shape Object shape, from 0 to 2. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

obj_size Object size. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

x X position of the sprite

y Y position of the sprite

3.24.2.9 static inline void PA_DualCreateSpriteFromGfx (u8 *obj_number*, u16 * *obj_gfx*, u8 *obj_shape*, u8 *obj_size*, u8 *color_mode*, u8 *palette*, s16 *x*, s16 *y*) [inline, static]

Create a sprite with it's gfx. This is the simple version of the function.

Parameters:

obj_number Object number you want to use (0-127 for each screen seperately).

obj_gfx Memory gfx to use. Get it by using PA_GetSpriteGfx or PA_CreateGfx

obj_shape Object shape, from 0 to 2. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

obj_size Object size. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

color_mode 256 or 16 color mode (1 or 0).

palette Palette to use (0-15).

x X position of the sprite

y Y position of the sprite

3.24.2.10 static inline void PA_DualCreateSpriteExFromGfx (u8 *obj_number*, u16 * *obj_gfx*, u8 *obj_shape*, u8 *obj_size*, u8 *color_mode*, u8 *palette*, u8 *obj_mode*, u8 *mosaic*, u8 *hflip*, u8 *vflip*, u8 *prio*, u8 *dblsize*, s16 *x*, s16 *y*) [inline, static]

Create a sprite with it's gfx. This is the complex version of the function.

Parameters:

obj_number Object number you want to use (0-127 for each screen seperately).

obj_gfx Memory gfx to use. Get it by using PA_GetSpriteGfx or PA_CreateGfx

obj_shape Object shape, from 0 to 2. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

obj_size Object size. Use the OBJ_SIZE_32X32 (...) macros for object shape and obj_size...

color_mode 256 or 16 color mode (1 or 0).

palette Palette to use (0-15).

obj_mode Object mode (normal, transparent, window). Not functionnal yet, please leave to 0 for now

mosaic Activate Mosaic for the sprite or not. Not yet functionnal either :p

hflip Horizontal flip on or off...

vflip Vertical flip...

prio Sprite priority regarding backgrounds : in front of which background to show it (0-3)

dblsize Double the possible sprite size. Activate only if you are going to rotate and zoom in the sprite

x X position of the sprite

y Y position of the sprite

3.24.2.11 static inline void PA_DualUpdateSpriteGfx (u8 *obj_number*, void * *obj_data*) [inline, static]

Update the Gfx of a given sprite.

Parameters:

obj_number Object number in the sprite system

obj_data Gfx to load

3.24.2.12 static inline void PA_DualUpdateGfx (u16 *gfx_number*, void * *obj_data*) [inline, static]

Update the Gfx of a given sprite.

Parameters:

gfx_number Gfx number in memory

obj_data Gfx to load

3.24.2.13 static inline void PA_DualDeleteSprite (u8 *obj_number*) [inline, static]

Delete a given sprite. If it is the only one to use it's gfx, it'll be deleted too.

Parameters:

obj_number Sprite number

3.24.2.14 static inline void PA_DualSetSpriteRotEnable (u8 *sprite*, u8 *rotset*) [inline, static]

Rotate and zoom a sprite.

Parameters:

sprite Sprite you want to rotate

rotset Rotset you want to give to that sprite (0-31). You can apparently use a rotset for multiple sprites if zoomed/rotated identically...

3.24.2.15 static inline void PA_DualSetSpriteRotDisable (u8 *sprite*) [inline, static]

Stop rotating and zooming a sprite.

Parameters:

sprite Sprite you want to rotate

3.24.2.16 static inline void PA_DualSetRotset (u8 *rotset*, s16 *angle*, u16 *zoomx*, u16 *zoomy*) [inline, static]

Rotate and zoom a sprite.

Parameters:

rotset Rotset you want to change. To give a sprite a rotset, use PA_SetSpriteRotEnable...

angle Angle, between 0 and 512 (not 360, be carefull)

zoomx Horizontal zoom. 256 is unzoomed, 512 is 2 times smaller, and 128 twice as big... So adjust at will ! :p

zoomy Vertical zoom. 256 is unzoomed, 512 is 2 times smaller, and 128 twice as big... So adjust at will ! :p

3.24.2.17 static inline void PA_DualSetRotsetNoZoom (u8 *rotset*, s16 *angle*) [inline, static]

Rotate a sprite without zooming. It's a bit faster than the normal PA_SetRotset function.

Parameters:

rotset Rotset you want to change. To give a sprite a rotset, use PA_SetSpriteRotEnable...

angle Angle, between 0 and 512 (not 360, be carefull)

3.24.2.18 static inline void PA_DualSetRotsetNoAngle (u8 *rotset*, u16 *zoomx*, u16 *zoomy*) [inline, static]

Zoom a sprite without rotating. It's a bit faster than the normal PA_SetRotset function.

Parameters:

rotset Rotset you want to change. To give a sprite a rotset, use PA_SetSpriteRotEnable...

zoomx Horizontal zoom. 256 is unzoomed, 512 is 2 times smaller, and 128 twice as big... So adjust at will ! :p

zoomy Vertical zoom. 256 is unzoomed, 512 is 2 times smaller, and 128 twice as big... So adjust at will ! :p

3.24.2.19 static inline void PA_DualSetSpritePal (u8 *obj*, u8 *pal*) [inline, static]

Set the color palette used by a sprite.

Parameters:

obj Object number in the sprite system

pal Palette number (0 - 15)

3.24.2.20 static inline void PA_DualSetSpriteDblsize (u8 *obj*, u8 *dblsize*) [inline, static]

Enable or disable double size for a given sprite.

Parameters:

obj Object number in the sprite system

dblsize 1 to enable doublesize, 0 to disable it...

3.24.2.21 static inline void PA_DualSetSpriteColors (u8 *sprite*, u8 *n_colors*) [inline, static]

Change the sprite's color mode.

Parameters:

sprite Object number in the sprite system

n_colors 0 for 16 colors, 1 for 256

3.24.2.22 static inline void PA_DualSetSpriteMode (u8 *sprite*, u8 *obj_mode*) [inline, static]

Set the sprite's mode : 0 for normal, 1 for alpha blending, 2 for window.

Parameters:

sprite Object number in the sprite system

obj_mode Object mode : 0 for normal, 1 for alpha blending, 2 for window ; not working yet

3.24.2.23 static inline void PA_DualSetSpriteMosaic (u8 *obj*, u8 *mosaic*) [inline, static]

Enable or disable mosaic mode for a given sprite.

Parameters:

obj Object number in the sprite system

mosaic Set mosaic on (1) or off (0)

3.24.2.24 static inline void PA_DualSetSpriteHflip (u8 *obj*, u8 *hflip*) [inline, static]

Enable or disable horizontal flip for a given sprite.

Parameters:

obj Object number in the sprite system

hflip Horizontal flip, 1 to enable, 0 to disable...

3.24.2.25 static inline void PA_DualSetSpriteVflip (u8 *obj*, u8 *vflip*) [inline, static]

Enable or disable vertical flip for a given sprite.

Parameters:

obj Object number in the sprite system

vflip Vertical flip, 1 to enable, 0 to disable...

3.24.2.26 static inline void PA_DualSetSpriteGfx (u8 *obj*, u16 * *gfx*) [inline, static]

Change the gfx used by a sprite.

Parameters:

obj Object number in the sprite system

gfx Gfx number ; you can get one by using PA_CreateGfx or **PA_GetSpriteGfx(obj_number)** (p. 116);

3.24.2.27 static inline void PA_DualSetSpritePrio (u8 *obj*, u8 *prio*) [inline, static]

Set a sprite's Background priority.

Parameters:

obj Object number in the sprite system

prio Sprite priority : 0 is over background 0, 1 over Bg 1, etc... (0-3)

3.24.2.28 static inline void PA_DualCloneSprite (u8 *obj*, u8 *target*) [inline, static]

Clone a sprite. Works only for sprites on the same screen.

Parameters:

obj Object number in the sprite system

target Target sprite to clone

3.24.2.29 static inline void PA_DualSetSpriteAnimEx (u8 *sprite*, u8 *lx*, u8 *ly*, u8 *ncolors*, s16 *animframe*) [inline, static]

Set the animation frame for a given sprite. This function is faster than the normal PA_SetSpriteAnim because it doesn't have to lookup the sprite dimensions...

Parameters:

sprite sprite number in the sprite system

lx Sprite width (8, 16, 32, 64)

ly Sprite height (8, 16, 32, 64)

ncolors Sprite color mode (0 for 16 colors, 1 for 256)

animframe Sprite animation frame (0, 1, 2, etc...)

3.24.2.30 static inline void PA_DualSetSpriteAnim (u8 *sprite*, s16 *animframe*) [inline, static]

Set the animation frame for a given sprite. Same as PA_SetSpriteAnimEx, but a bit slower and easier to use...

Parameters:

sprite sprite number in the sprite system

animframe Sprite animation frame (0, 1, 2, etc...)

3.24.2.31 static inline void PA_DualStartSpriteAnimEx (u8 *sprite*, s16 *firstframe*, s16 *lastframe*, s16 *speed*, u8 *type*, s16 *ncycles*) [inline, static]

Start a sprite animation for DualSprites. Once started, it continues on and on by itself until you stop it !

Parameters:

sprite sprite number in the sprite system

firstframe First frame of the animation sequence, most of the time 0...

lastframe Last frame to be displayed. When it gets there, it loops back to the first frame

speed Speed, in frames per second. So speed 1 would mean 1 image per second, so 1 image every game frame

type Defines how you want it to loop. ANIM_LOOP (0) for a normal loop, ANIM_UPDOWN (1) for back and forth animation.

ncycles Number of animation cycles before stopping. If using ANIM_UPDOWN, it takes 2 cycles to come back to the original image

3.24.2.32 static inline void PA_DualStartSpriteAnim (u8 *sprite*, s16 *firstframe*, s16 *lastframe*, s16 *speed*) [inline, static]

Start a sprite animation for DualSprite. Once started, it continues on and on by itself until you stop it !

Parameters:

sprite sprite number in the sprite system

firstframe First frame of the animation sequence, most of the time 0...

lastframe Last frame to be displayed. When it gets there, it loops back to the first frame

speed Speed, in frames per second. So speed 1 would mean 1 image per second, so 1 image every game frame

3.24.2.33 static inline void PA_DualStopSpriteAnim (u8 *sprite*) [inline, static]

Stop a sprite animation for DualSprites.

Parameters:

sprite sprite number in the sprite system

3.24.2.34 static inline void PA_DualSetSpriteAnimFrame (u8 *sprite*, u16 *frame*) [inline, static]

Set the current animation frame number for DualSprites.

Parameters:

sprite sprite number in the sprite system

frame Frame number to use...

3.24.2.35 static inline u16 PA_DualGetSpriteAnimFrame (u8 *sprite*) [inline, static]

Returns the current animation frame number for DualSprites.

Parameters:

sprite sprite number in the sprite system

3.24.2.36 static inline void PA_DualSetSpriteAnimSpeed (u8 *sprite*, s16 *speed*) [inline, static]

Set the current animation speed for DualSprites.

Parameters:

sprite sprite number in the sprite system

speed Speed, in fps...

3.24.2.37 static inline u16 PA_DualGetSpriteAnimSpeed (u8 *sprite*) [inline, static]

Returns the current animation speed for DualSprites.

Parameters:

sprite sprite number in the sprite system

3.24.2.38 static inline void PA_DualSpriteAnimPause (u8 *sprite*, u8 *pause*) [inline, static]

Pause or UnPause a sprite animation for DualSprites.

Parameters:

sprite sprite number in the sprite system

pause 1 for pause, 0 for unpause

3.25 Text output system

Defines

- #define **PA_InitText** PA_LoadDefaultText

  *Old name for **PA_LoadDefaultText()** (p. 147).*

- #define **PA_SetTileLetter**(screen, x, y, letter) PA_SetMapTileAll(screen, PAbgtext[screen], x, y, (PA_textmap[screen][(u16)letter]&((1<<12)-1)) + (PAtext_pal[screen] << 12))

  Output a letter on the DS screen.

- #define **PA_InitCustomText**(screen, bg_select, text) PA_InitCustomTextEx(screen, bg_select, text##_Tiles, text##_Map, text##_Pal)

  [DEPRECATED] Init the text using one of your own fonts !

- #define **PA_ShowFont**(screen) PA_LoadBgMap(screen, PAbgtext[screen], (void*)PA_textmap[screen], BG_256X256)

  Show the current font used. This is just for debug, no real use ingame.

- #define **PA_8bitCustomFont**(bit8_slot, bit8_font)

  [DEPRECATED] Add custom fonts to the 8bit Font system !! Font must be converted with PAGfx

Functions

- void **PA_LoadDefaultText** (u8 screen, u8 bg_select)

  Load and initialize the default text. Works only in modes 0-2.

- static void **PA_SetTextTileCol** (u8 screen, u8 color)

  Change the text writing color (does not change the current text's color).

- void **PA_OutputText** (u8 screen, u16 x, u16 y, const char *text,...)

  Output text on the DS screen. Works only in modes 0-2.

- u16 **PA_OutputSimpleText** (u8 screen, u16 x, u16 y, const char *text)

  Output simple text on the DS screen. Works only in modes 0-2. Much faster than PA_OutputText, but much more limited... Returns the number of letters.

- u32 **PA_BoxText** (u8 screen, u16 basex, u16 basey, u16 maxx, u16 maxy, const char *text, u32 limit)

  Output text on the DS screen. This text is limited to a chosen box, and you can chose the number of letters to output (can be used to show 'typed' text, just put 10000 if you want to show all the text...). Returns the number of letters outputed.

- u32 **PA_BoxTextNoWrap** (u8 screen, u16 basex, u16 basey, u16 maxx, u16 maxy, const char *text, u32 limit)

  Output text on the DS screen. This text is limited to a chosen box, and you can chose the number of letters to output (can be used to show 'typed' text, just put 10000 if you want to show all the text...). Returns the number of letters outputed. This function does not support word wrapping.

- static void **PA_SetTextCol** (u8 screen, u16 r, u16 g, u16 b)

  Change the screen text's default color.

- void **PA_LoadText** (u8 screen, u8 bg_number, const **PA_BgStruct** *font)

  Load and initialize a custom font.

- s16 **PA_8bitText** (u8 screen, s16 basex, s16 basey, s16 maxx, s16 maxy, const char *text, u8 color, u8 size, u8 transp, s32 limit)

  This is a variable width and variable size function to draw text on the screen. It draws on an 8 bit background (see PA_Init8bitBg for more info), and has options such as size, transaprency, and box limits, as well as the color. Only problem : it does not take commands such as d, etc... The function returns the number of characters it outputed.

- s16 **PA_CenterSmartText** (u8 screen, s16 basex, s16 basey, s16 maxx, s16 maxy, const char *text, u8 color, u8 size, u8 transp)

  Basicaly the same as the SmartText function, but this time centered...

- void **PA_AddBitmapFont** (int slot, const **PA_BgStruct** *font)

  Add a custom font to the 8bit/16bit font system.

- void **PA_InitTextBorders** (u8 screen, u8 x1, u8 y1, u8 x2, u8 y2)

  Initialise a text box with it's borders. This makes writing in a delimited area much easier...

- void **PA_EraseTextBox** (u8 screen)

  Erases the text in a textbox. Requires that that box be initialized with PA_InitTextBorders.

- static u32 **PA_SimpleBoxText** (u8 screen, const char *text, u32 limit)

  Write text in an initiliazed textbox. Similar to PA_BoxText, but without needing the text limits.

- void **PA_ClearTextBg** (u8 screen)

  Erase all the text on a given screen.

- void **PA_Print** (u8 screen, const char *text,...)

  Output text on the DS screen. Works like a printf function.

- static void **PA_PrintLetter** (u8 screen, char letter)

  Like PA_Print, but for a letter.

3.25.1 Detailed Description

Allows you to output text...

3.25.2 Define Documentation

3.25.2.1 #define PA_SetTileLetter(screen, x, y, letter) PA_SetMapTileAll(screen, PAbgtext[screen], x, y, (PA_textmap[screen][(u16)letter]&((1<<12)-1)) + (PAtext_pal[screen] << 12))

Output a letter on the DS screen.

Parameters:

screen Chose de screen (0 or 1)

x X coordinate in TILES (0-31) where to write the letter

y Y coordinate in TILES (0-19) where to write the letter

letter Letter... 'a', 'Z', etc...

3.25.2.2 #define PA_InitCustomText(screen, bg_select, text) PA_InitCustomTextEx(screen, bg_select, text##_Tiles, text##_Map, text##_Pal)

[DEPRECATED] Init the text using one of your own fonts !

Deprecated

Parameters:

screen Chose de screen (0 or 1)

bg_select Background number...

text Font image file name converted with PAGfx

3.25.2.3 #define PA_ShowFont(screen) PA_LoadBgMap(screen, PAbgtext[screen], (void*)PA_textmap[screen], BG_256X256)

Show the current font used. This is just for debug, no real use ingame.

Parameters:

screen Chose de screen (0 or 1)

3.25.2.4 #define PA_8bitCustomFont(bit8_slot, bit8_font)

Value:

```
do{\
    PA_DEPRECATED_MACRO;\
    bittext_maps[bit8_slot] = (u16*)(void*)bit8_font##_Map; \
    bit8_tiles[bit8_slot] = (u8*)bit8_font##_Tiles; \
    pa_bittextdefaultsize[bit8_slot] = (u8*)bit8_font##_Sizes;  \
    pa_bittextpoliceheight[bit8_slot] = bit8_font##_Height;\
}while(0)
```

[DEPRECATED] Add custom fonts to the 8bit Font system !! Font must be converted with PAGfx

Deprecated

Parameters:

bit8_slot Font slot... 0-4 are used by the defaut PAlib fonts, 5-9 are free to use. You can freely overwrite the PAlib fonts if you want

bit8_font Font name;..

3.25.3 Function Documentation

3.25.3.1 void PA_LoadDefaultText (u8 *screen*, u8 *bg_select*)

Load and initialize the default text. Works only in modes 0-2.

Parameters:

screen Choose the screen (0 or 1)

bg_select Background number (0-3)

Examples:

Backgrounds/Effects/Mode7/source/main.c, and **Text/Normal/HelloWorld/-source/main.c**.

3.25.3.2 static inline void PA_SetTextTileCol (u8 *screen*, u8 *color*) [inline, static]

Change the text writing color (does not change the current text's color).

Parameters:

screen Chose de screen (0 or 1)

color Color, from 0 to 6, just test to see the result...

3.25.3.3 void PA_OutputText (u8 *screen*, u16 *x*, u16 *y*, const char * *text*, ...)

Output text on the DS screen. Works only in modes 0-2.

Parameters:

screen Chose de screen (0 or 1)

x X coordinate in TILES (0-31) where to begin writing the text

y Y coordinate in TILES (0-19) where to begin writing the text

text String to output. The following commands are avaiblable : %s to output another string, %d to output a value, %fX to output a float with X digits, \n to go to the line. Here's an example : PA_OutputText(0, 0, 1, "My name is %s and I have only %d teeth", "Mollusk", 20);

Examples:

Backgrounds/Effects/Mode7/source/main.c.

3.25.3.4 u16 PA_OutputSimpleText (u8 *screen*, u16 *x*, u16 *y*, const char * *text*)

Output simple text on the DS screen. Works only in modes 0-2. Much faster than PA_OutputText, but much more limited... Returns the number of letters.

Parameters:

screen Chose de screen (0 or 1)

x X coordinate in TILES (0-31) where to begin writing the text

y Y coordinate in TILES (0-19) where to begin writing the text

text String to output.

Examples:

Text/Normal/HelloWorld/source/main.c.

3.25.3.5 u32 PA_BoxText (u8 *screen*, u16 *basex*, u16 *basey*, u16 *maxx*, u16 *maxy*, const char * *text*, u32 *limit*)

Output text on the DS screen. This text is limited to a chosen box, and you can chose the number of letters to output (can be used to show 'typed' text, just put 10000 if you want to show all the text...). Returns the number of letters outputed.

Parameters:

screen Chose de screen (0 or 1)

basex X coordinate in TILES (0-31) where to begin writing the text

basey Y coordinate in TILES (0-19) where to begin writing the text

maxx X coordinate in TILES (0-31) where to stop writing the text

maxy Y coordinate in TILES (0-19) where to stop writing the text

text String to output.

limit Maximum number of letters to show this time

3.25.3.6 u32 PA_BoxTextNoWrap (u8 *screen*, u16 *basex*, u16 *basey*, u16 *maxx*, u16 *maxy*, const char * *text*, u32 *limit*)

Output text on the DS screen. This text is limited to a chosen box, and you can chose the number of letters to output (can be used to show 'typed' text, just put 10000 if you want to show all the text...). Returns the number of letters outputed. This function does not support word wrapping.

Parameters:

screen Chose de screen (0 or 1)

basex X coordinate in TILES (0-31) where to begin writing the text

basey Y coordinate in TILES (0-19) where to begin writing the text

maxx X coordinate in TILES (0-31) where to stop writing the text

maxy Y coordinate in TILES (0-19) where to stop writing the text

text String to output.

limit Maximum number of letters to show this time

3.25.3.7 static inline void PA_SetTextCol (u8 *screen*, u16 *r*, u16 *g*, u16 *b*) [inline, static]

Change the screen text's default color.

Parameters:

screen Chose de screen (0 or 1)

r Red amount (0-31)

g Green amount (0-31)

b Blue amount (0-31)

3.25.3.8 void PA_LoadText (u8 *screen*, u8 *bg_select*, const PA_BgStruct * *font*)

Load and initialize a custom font.

Parameters:

screen Chose the screen (0 or 1)

bg_select Background number...

font Pointer to the font

3.25.3.9 s16 PA_8bitText (u8 *screen*, s16 *basex*, s16 *basey*, s16 *maxx*, s16 *maxy*, const char * *text*, u8 *color*, u8 *size*, u8 *transp*, s32 *limit*)

This is a variable width and variable size function to draw text on the screen. It draws on an 8 bit background (see PA_Init8bitBg for more info), and has options such as size, transaprency, and box limits, as well as the color. Only problem : it does not take commands such as d, etc... The function returns the number of characters it outputed.

Parameters:

screen Chose de screen (0 or 1)

basex X coordinate of the top left corner

basey Y coordinate of the top left corner

maxx X coordinate of the down right corner

maxy Y coordinate of the down right corner

text Text, such as "Hello World"

color Palette color to use (0-255)

size Size of the text, from 0 (really small) to 4 (pretty big)

transp Transparency. Setting this to 0 will overwrite all drawing in the text zone. 1 will write the text without erasing the drawing. 2 won't output anything (just to count the letters), 3 is rotated one way, 4 rotated the other way

limit You can give a maximum number of characters to output. This can be usefull to have a slowing drawing text (allow to draw 1 more character each frame...)

3.25.3.10 s16 PA_CenterSmartText (u8 *screen*, s16 *basex*, s16 *basey*, s16 *maxx*, s16 *maxy*, const char * *text*, u8 *color*, u8 *size*, u8 *transp*)

Basicaly the same as the SmartText function, but this time centered...

Parameters:

screen Chose de screen (0 or 1)

basex X coordinate of the top left corner

basey Y coordinate of the top left corner

maxx X coordinate of the down right corner

maxy Y coordinate of the down right corner

text Text, such as "Hello World"

color Palette color to use (0-255)

size Size of the text, from 0 (really small) to 4 (pretty big)

transp Transparency. Setting this to 0 will overwrite all drawing in the text zone. 1 will write the text without erasing the drawing. 2 won't output anything (just to count the letters), 3 is rotated one way, 4 rotated the other way

3.25.3.11 void PA_AddBitmapFont (int *slot*, const PA_BgStruct * *font*)

Add a custom font to the 8bit/16bit font system.

Parameters:

slot Font slot. 0-4 are used by the default PAlib fonts, 5-9 are free to use. You can freely overwrite the PAlib fonts if you want.

font Pointer to the font.

3.25.3.12 void PA_InitTextBorders (u8 *screen*, u8 *x1*, u8 *y1*, u8 *x2*, u8 *y2*)

Initialise a text box with it's borders. This makes writing in a delimited area much easier...

Parameters:

screen Chose de screen (0 or 1)

x1 Left limit in tiles

y1 Top

x2 Right

y2 Bottom

3.25.3.13 void PA_EraseTextBox (u8 *screen*)

Erases the text in a textbox. Requires that that box be initialized with PA_-InitTextBorders.

Parameters:

screen Chose de screen (0 or 1)

3.25.3.14 static inline u32 PA_SimpleBoxText (u8 *screen*, const char * *text*, u32 *limit*) `[inline, static]`

Write text in an initiliazed textbox. Similar to PA_BoxText, but without needing the text limits.

Parameters:

screen Chose de screen (0 or 1)

text String to output.

limit Maximum number of letters to show this time

3.25.3.15 void PA_ClearTextBg (u8 *screen*)

Erase all the text on a given screen.

Parameters:

screen Chose de screen (0 or 1)

3.25.3.16 void PA_Print (u8 *screen*, const char * *text*, ...)

Output text on the DS screen. Works like a printf function.

Parameters:

screen Chose de screen (0 or 1)

text String to output. The following commands are avaiblable : %s to output another string, %d to output a value, %fX to output a float with X digits, \n to go to the line. Here's an example : PA_OutputText(0, 0, 1, "My name is %s and I have only %d teeth", "Mollusk", 20);

3.25.3.17 static inline void PA_PrintLetter (u8 *screen*, char *letter*) [inline, static]

Like PA_Print, but for a letter.

Parameters:

screen Chose de screen (0 or 1)

letter Any letter...

3.26 Bg Modes on 2 Screens

Defines

- #define **PA_DualLoadTiledBg**(bg_number, bg_name)

  [DEPRECATED] This will never get easier... Loads a background TiledBg converted with PAGfx, with it's tiles, map, and palette. Only 256 color mode available. On 2 screens as 1...

- #define **PA_DualLoadSimpleBg**(bg_select, bg_tiles, bg_map, bg_size, wraparound, color_mode)

  [DEPRECATED] Simplest way to load a Background on both screens

- #define **PA_DualLoadRotBg**(bg_select, bg_tiles, bg_map, bg_size, wraparound)

  [DEPRECATED] Load a background fit for rotating/scaling ! Warning, you must use PA_SetVideoMode to 1 if you want 1 rotating background (Bg3 only !), or 2 for 2 rotating backgrounds (Bg2 and 3). The background MUST be in 256 colors

- #define **PA_DualLoadBg**(bg_select, bg_tiles, tile_size, bg_map, bg_size, wraparound, color_mode)

  [DEPRECATED] Simplest way to load a Background. Combines PA_InitBg, PA_LoadBgTiles, and PA_LoadBgMap

- #define **PA_DualLoadPAGfxLargeBg**(bg_number, bg_name)

  [DEPRECATED] Completely load and initialise a background with infinite scrolling (usefull if larger or wider than 512 pixels), converted with PAGfx. Background on both screens, as one

- #define **PA_DualLoadLargeBg**(bg_select, bg_tiles, bg_map, color_mode, lx, ly)

  [DEPRECATED] Completely load and initialise a background with infinite scrolling (usefull if larger or wider than 512 pixels), on both screens

- #define **PA_DualLoadLargeBgEx**(bg_select, bg_tiles, tile_size, bg_map, color_mode, lx, ly)

  [DEPRECATED] Completely load and initialise a background with infinite scrolling (usefull if larger or wider than 512 pixels), but here you can put yourself the tile size...

- #define **PA_DualEasyBgLoad**(bg_number, bg_name)

  [DEPRECATED] EasyBg load, but for Dual screen...

Functions

- static void **PA_DualHideBg** (u8 bg_select)

  Hide a background on both screens.

- static void **PA_DualShowBg** (u8 bg_select)

  Show a hidden background, on both screens.

- static void **PA_DualResetBg** (void)

  Reinitialize de Bg system.

- static void **PA_DualDeleteBg** (u8 bg_select)

  Delete a complete background (tiles + map + hide it...).

- static void **PA_DualBGScrollX** (u8 bg_number, s16 x)

  Scroll horizontaly any background, on both screens.

- static void **PA_DualBGScrollY** (u8 bg_number, s16 y)

  Scroll vertically any background.

- static void **PA_DualBGScrollXY** (u8 bg_number, s16 x, s16 y)

  Scroll horizontaly and vertically any background.

- static void **PA_DualEasyBgScrollX** (u8 bg_select, s32 x)

  Scroll an EasyBg horizontaly. It must have been initialised with PA_LoadLargeBg.

- static void **PA_DualEasyBgScrollY** (u8 bg_select, s32 y)

  Scroll an EasyBg vertically.

- static void **PA_DualLoadBackground** (u8 bg_number, const **PA_BgStruct** *bg)

  Load a background (EasyBg, RotBg or UnlimitedBg), but for Dual screen...

- static void **PA_DualEasyBgScrollXY** (u8 bg_select, s32 x, s32 y)

  Scroll a Dual EasyBg.

- static void **PA_DualInfLargeScrollX** (u8 bg_select, s32 x)

  Scroll a large infinite scrolling background horizontaly. It must have been initialised with PA_LoadLargeBg.

- static void **PA_DualInfLargeScrollY** (u8 bg_select, s32 y)

  Scroll a large infinite scrolling background vertically. It must have been initialised with PA_LoadLargeBg.

- static void **PA_DualInfLargeScrollXY** (u8 bg_select, s32 x, s32 y)

  Scroll a large infinite scrolling background horizontaly and vertically. It must have been initialised with PA_LoadLargeBg.

- static void **PA_DualLargeScrollX** (u8 bg_select, s32 x)

Scroll a large background horizontaly. It must have been initialised with PA_-LoadLargeBg. This function does not wrap around, but is faster than the InfLarge-Scroll...

- static void **PA_DualLargeScrollY** (u8 bg_select, s32 y)

  Scroll a large background vertically. It must have been initialised with PA_-LoadLargeBg. This function does not wrap around, but is faster than the InfLarge-Scroll...

- static void **PA_DualLargeScrollXY** (u8 bg_select, s32 x, s32 y)

  Scroll a large background horizontaly and vertically. It must have been initialised with PA_LoadLargeBg. This function does not wrap around, but is faster than the InfLargeScroll...

- static void **PA_DualInitParallaxX** (s32 bg0, s32 bg1, s32 bg2, s32 bg3)

  Initialise Parallax Scrolling for multiple backgrounds, horizontaly. Chose the speed at which each background will scroll compared to the others. Then use PA_-ParallaxScrollX to scroll...

- static void **PA_DualInitParallaxY** (s32 bg0, s32 bg1, s32 bg2, s32 bg3)

  Initialise Parallax Scrolling for multiple backgrounds, horizontaly. Chose the speed at which each background will scroll compared to the others. Then use PA_-ParallaxScrollX to scroll...

- static void **PA_DualParallaxScrollX** (s32 x)

  Scroll the backgrounds.

- static void **PA_DualParallaxScrollY** (s32 y)

  Scroll the backgrounds.

- static void **PA_DualParallaxScrollXY** (s32 x, s32 y)

  Scroll the backgrounds.

- static void **PA_DualSetBgPrio** (u8 bg, u8 prio)

  Change a backgrounds priority.

3.26.1 Detailed Description

Load tiles, a map, scroll it... and 2 screens automatically

3.26.2 Define Documentation

3.26.2.1 #define PA_DualLoadTiledBg(bg_number, bg_name)

Value:

```
do{\
    PA_DEPRECATED_MACRO;\
    PA_LoadTiledBg(0, bg_number, bg_name);\
    PA_LoadTiledBg(1, bg_number, bg_name);\
    PA_DualBGScrollY(bg_number, 0);}while(0)
```

[DEPRECATED] This will never get easier... Loads a background TiledBg converted with PAGfx, with it's tiles, map, and palette. Only 256 color mode available. On 2 screens as 1...

Deprecated

Parameters:

bg_number Background number to load (from 0 to 3)

bg_name Background name, like bg0

3.26.2.2 #define PA_DualLoadSimpleBg(bg_select, bg_tiles, bg_map, bg_size, wraparound, color_mode)

Value:

```
do{\
    PA_DEPRECATED_MACRO;\
    PA_LoadSimpleBg(0, bg_select, bg_tiles, bg_map, bg_size, wraparound, color_mo
      de);\
    PA_LoadSimpleBg(1, bg_select, bg_tiles, bg_map, bg_size, wraparound, color_mo
      de);\
    PA_DualBGScrollY(bg_select, 0);}while(0)
```

[DEPRECATED] Simplest way to load a Background on both screens

Deprecated

Parameters:

bg_select Background number to load (from 0 to 3)

bg_tiles Name of the tiles' info (example: ship_Tiles)

bg_map Name of the map's info (example : ship_Map)

bg_size Background size. To use a normal background, use the macros BG_-256X256, BG_256X512, etc...

wraparound If the background wraps around or not. More important for rotating backgrounds.

color_mode Color mode : 0 for 16 color mode, 1 for 256...

3.26.2.3 #define PA_DualLoadRotBg(bg_select, bg_tiles, bg_map, bg_size, wraparound)

Value:

```
do{\
    PA_DEPRECATED_MACRO;\
    PA_LoadRotBg(0, bg_select, bg_tiles, bg_map, bg_size, wraparound);\
    PA_LoadRotBg(1, bg_select, bg_tiles, bg_map, bg_size, wraparound);\
    PA_DualBGScrollY(bg_select, 0);}while(0)
```

[DEPRECATED] Load a background fit for rotating/scaling ! Warning, you must use PA_SetVideoMode to 1 if you want 1 rotating background (Bg3 only !), or 2 for 2 rotating backgrounds (Bg2 and 3). The background MUST be in 256 colors

Deprecated

Parameters:

bg_select Background number to load

bg_tiles Name of the tiles' info (example: ship_Tiles)

bg_map Name of the map's info (example : ship_Map)

bg_size Background size. Use the following macros : BG_ROT_128X128, or 256X256, 512X512, or 1024X1024

wraparound If the background wraps around or not.

3.26.2.4 #define PA_DualLoadBg(bg_select, bg_tiles, tile_size, bg_map, bg_size, wraparound, color_mode)

Value:

```
do{\
    PA_DEPRECATED_MACRO;\
    PA_LoadBg(0, bg_select, bg_tiles, tile_size, bg_map, bg_size, wraparound, col
      or_mode);\
    PA_LoadBg(1, bg_select, bg_tiles, tile_size, bg_map, bg_size, wraparound, col
      or_mode);\
    PA_DualBGScrollY(bg_select, 0);}while(0)
```

[DEPRECATED] Simplest way to load a Background. Combines PA_InitBg, PA_-LoadBgTiles, and PA_LoadBgMap

Deprecated

Parameters:

bg_select Background number to load (from 0 to 3)

bg_tiles Name of the tiles' info (example: ship_Tiles)

tile_size Size of your tileset

bg_map Name of the map's info (example : ship_Map)

bg_size Background size. This is important, because it also determines whether the Bg is rotatable or not. To use a normal background, use the macros BG_256X256, BG_256X512, etc... For a rotatable Bg, use the macros BG_-ROT_128X128...

wraparound If the background wraps around or not. More important for rotating backgrounds.

color_mode Color mode : 0 for 16 color mode, 1 for 256...

3.26.2.5 #define PA_DualLoadPAGfxLargeBg(bg_number, bg_name)

Value:

```
do{\
    PA_DEPRECATED_MACRO;\
    PA_LoadPAGfxLargeBg(0, bg_number, bg_name);\
    PA_LoadPAGfxLargeBg(1, bg_number, bg_name);\
    PA_DualInfLargeScrollY(bg_number, 0);}while(0)
```

[DEPRECATED] Completely load and initialise a background with infinite scrolling (usefull if larger or wider than 512 pixels), converted with PAGfx. Background on both screens, as one

Deprecated

Parameters:

bg_number Background number to load (from 0 to 3)

bg_name Background name, in PAGfx

3.26.2.6 #define PA_DualLoadLargeBg(bg_select, bg_tiles, bg_map, color_mode, lx, ly)

Value:

```
do{\
    PA_DEPRECATED_MACRO;\
    PA_LoadLargeBg(0, bg_select, bg_tiles, bg_map, color_mode, lx, ly);\
    PA_LoadLargeBg(1, bg_select, bg_tiles, bg_map, color_mode, lx, ly);\
    PA_DualInfLargeScrollY(bg_select, 0);}while(0)
```

[DEPRECATED] Completely load and initialise a background with infinite scrolling (usefull if larger or wider than 512 pixels), on both screens

Deprecated

Parameters:

bg_select Background number to load (from 0 to 3)

bg_tiles Name of the tiles' info (example: ship_Tiles)

bg_map Name of the map's info (example : ship_Map)

color_mode Color mode : 0 for 16 color mode, 1 for 256...

lx Width, in tiles. So a 512 pixel wide map is 64 tiles wide...

ly Height, in tiles. So a 512 pixel high map is 64 tiles high...

3.26.2.7 #define PA_DualLoadLargeBgEx(bg_select, bg_tiles, tile_size, bg_map, color_mode, lx, ly)

Value:

```
do{\
    PA_DEPRECATED_MACRO;\
    PA_LoadLargeBgEx(0, bg_select, bg_tiles, tile_size, bg_map, color_mode, lx, l
      y);\
    PA_LoadLargeBgEx(1, bg_select, bg_tiles, tile_size, bg_map, color_mode, lx, l
      y);\
    PA_DualInfLargeScrollY(bg_select, 0);}while(0)
```

[DEPRECATED] Completely load and initialise a background with infinite scrolling (usefull if larger or wider than 512 pixels), but here you can put yourself the tile size...

Deprecated

Parameters:

bg_select Background number to load (from 0 to 3)

bg_tiles Name of the tiles' info (example: ship_Tiles)

tile_size Size of your tileset

bg_map Name of the map's info (example : ship_Map)

color_mode Color mode : 0 for 16 color mode, 1 for 256...

lx Width, in tiles. So a 512 pixel wide map is 64 tiles wide...

ly Height, in tiles. So a 512 pixel high map is 64 tiles high...

3.26.2.8 #define PA_DualEasyBgLoad(bg_number, bg_name)

Value:

```
do{\
    PA_DEPRECATED_MACRO;\
    PA_EasyBgLoad(0, bg_number, bg_name);\
    PA_EasyBgLoad(1, bg_number, bg_name);\
    PA_DualEasyBgScrollY(bg_number, 0);}while(0)
```

[DEPRECATED] EasyBg load, but for Dual screen...

Deprecated

Parameters:

bg_number Background number to load (from 0 to 3)

bg_name Background name, in PAGfx

3.26.3 Function Documentation

3.26.3.1 static inline void PA_DualHideBg (u8 *bg_select*) [inline, static]

Hide a background on both screens.

Parameters:

bg_select Background number to load (from 0 to 3)

3.26.3.2 static inline void PA_DualShowBg (u8 *bg_select*) [inline, static]

Show a hidden background, on both screens.

Parameters:

bg_select Background number to load (from 0 to 3)

3.26.3.3 static inline void PA_DualDeleteBg (u8 *bg_select*) [inline, static]

Delete a complete background (tiles + map + hide it...).

Parameters:

bg_select Background number to load (from 0 to 3)

3.26.3.4 static inline void PA_DualBGScrollX (u8 *bg_number*, s16 *x*) [inline, static]

Scroll horizontaly any background, on both screens.

Parameters:

bg_number Background number (0-3)

x X value to scroll

3.26.3.5 static inline void PA_DualBGScrollY (u8 *bg_number*, s16 *y*) [inline, static]

Scroll vertically any background.

Parameters:

bg_number Background number (0-3)

y Y value to scroll

3.26.3.6 static inline void PA_DualBGScrollXY (u8 *bg_number*, s16 *x*, s16 *y*) [inline, static]

Scroll horizontaly and vertically any background.

Parameters:

bg_number Background number (0-3)

x X value to scroll

y Y value to scroll

3.26.3.7 static inline void PA_DualEasyBgScrollX (u8 *bg_select*, s32 *x*) [inline, static]

Scroll an EasyBg horizontaly. It must have been initialised with PA_LoadLargeBg.

Parameters:

bg_select Background number to load (from 0 to 3)

x X value to scroll

3.26.3.8 static inline void PA_DualEasyBgScrollY (u8 *bg_select*, s32 *y*) [inline, static]

Scroll an EasyBg vertically.

Parameters:

bg_select Background number to load (from 0 to 3)

y Y value to scroll

3.26.3.9 static inline void PA_DualLoadBackground (u8 *bg_number*, const PA_BgStruct * *bg*) [inline, static]

Load a background (EasyBg, RotBg or UnlimitedBg), but for Dual screen...

Parameters:

bg_number Background number to load (from 0 to 3)

bg Pointer to the background (struct)

3.26.3.10 static inline void PA_DualEasyBgScrollXY (u8 *bg_select*, s32 *x*, s32 *y*) [inline, static]

Scroll a Dual EasyBg.

Parameters:

bg_select Background number to load (from 0 to 3)

x X value to scroll

y Y value to scroll

3.26.3.11 static inline void PA_DualInfLargeScrollX (u8 *bg_select*, s32 *x*) [inline, static]

Scroll a large infinite scrolling background horizontaly. It must have been initialised with PA_LoadLargeBg.

Parameters:

bg_select Background number to load (from 0 to 3)

x X value to scroll

3.26.3.12 static inline void PA_DualInfLargeScrollY (u8 *bg_select*, s32 *y*) [inline, static]

Scroll a large infinite scrolling background vertically. It must have been initialised with PA_LoadLargeBg.

Parameters:

bg_select Background number to load (from 0 to 3)

y Y value to scroll

3.26.3.13 static inline void PA_DualInfLargeScrollXY (u8 *bg_select*, s32 *x*, s32 *y*) [inline, static]

Scroll a large infinite scrolling background horizontaly and vertically. It must have been initialised with PA_LoadLargeBg.

Parameters:

bg_select Background number to load (from 0 to 3)

x X value to scroll

y Y value to scroll

3.26.3.14 static inline void PA_DualLargeScrollX (u8 *bg_select*, s32 *x*) [inline, static]

Scroll a large background horizontaly. It must have been initialised with PA_LoadLargeBg. This function does not wrap around, but is faster than the InfLargeScroll...

Parameters:

bg_select Background number to load (from 0 to 3)

x X value to scroll

3.26.3.15 static inline void PA_DualLargeScrollY (u8 *bg_select*, s32 *y*) [inline, static]

Scroll a large background vertically. It must have been initialised with PA_LoadLargeBg. This function does not wrap around, but is faster than the InfLargeScroll...

Parameters:

bg_select Background number to load (from 0 to 3)

y Y value to scroll

3.26.3.16 static inline void PA_DualLargeScrollXY (u8 *bg_select*, s32 *x*, s32 *y*) [inline, static]

Scroll a large background horizontaly and vertically. It must have been initialised with PA_LoadLargeBg. This function does not wrap around, but is faster than the InfLargeScroll...

Parameters:

bg_select Background number to load (from 0 to 3)

x X value to scroll

y Y value to scroll

3.26.3.17 static inline void PA_DualInitParallaxX (s32 *bg0*, s32 *bg1*, s32 *bg2*, s32 *bg3*) [inline, static]

Initialise Parallax Scrolling for multiple backgrounds, horizontaly. Chose the speed at which each background will scroll compared to the others. Then use PA_-ParallaxScrollX to scroll...

Parameters:

- ***bg0*** Value for the first background (0). Set to 256 for normal scroll speed, lower for lower speed (128 is half speed...), higher for faster (512 is twice as fast...). You can set negative values. 0 inactivates parallax scrolling for this background
- ***bg1*** Same thing for Background 1
- ***bg2*** Same thing for Background 2
- ***bg3*** Same thing for Background 3

3.26.3.18 static inline void PA_DualInitParallaxY (s32 *bg0*, s32 *bg1*, s32 *bg2*, s32 *bg3*) [inline, static]

Initialise Parallax Scrolling for multiple backgrounds, horizontaly. Chose the speed at which each background will scroll compared to the others. Then use PA_-ParallaxScrollX to scroll...

Parameters:

- ***bg0*** Value for the first background (0). Set to 256 for normal scroll speed, lower for lower speed (128 is half speed...), higher for faster (512 is twice as fast...). You can set negative values. 0 inactivates parallax scrolling for this background
- ***bg1*** Same thing for Background 1
- ***bg2*** Same thing for Background 2
- ***bg3*** Same thing for Background 3

3.26.3.19 static inline void PA_DualParallaxScrollX (s32 *x*) [inline, static]

Scroll the backgrounds.

Parameters:

- ***x*** X value to scroll

3.26.3.20 static inline void PA_DualParallaxScrollY (s32 *y*) [inline, static]

Scroll the backgrounds.

Parameters:

y Y value to scroll

3.26.3.21 static inline void PA_DualParallaxScrollXY (s32 *x*, s32 *y*) [inline, static]

Scroll the backgrounds.

Parameters:

x X value to scroll

y Y value to scroll

3.26.3.22 static inline void PA_DualSetBgPrio (u8 *bg*, u8 *prio*) [inline, static]

Change a backgrounds priority.

Parameters:

bg Background...

prio Priority level (0-3, 0 being the highest)

3.27 Window system

Defines

- #define **PA_SetWin1XY**(screen, x1, y1, x2, y2) do{WIN1X(screen) = x2 + ((x1) << 8); WIN1Y(screen) = y2 + ((y1) << 8);}while(0)

  Set the X et Y coordinates of the rectangular second window. You'll also have to use PA_SetWin1 to chose which Backgrounds are visible and if sprites are too...

- #define **PA_EnableWin0**(screen, bg_sprites) do{DISPCNTL(screen) |= WINDOW0; WININ(screen) &= 255; WININ(screen) |= (bg_sprites);}while(0)

  Enable and set which backgrounds will be visible and whether sprites will too or not, for Window 0. You'll then have to configure it with PA_SetWin0XY.

- #define **PA_DisableWin0**(screen) DISPCNTL(screen) &= ∼WINDOW0

  Disable the first window...

- #define **PA_EnableWin1**(screen, bg_sprites) do{DISPCNTL(screen) |= WINDOW1; WININ(screen) &= 255; WININ(screen) |= ((bg_sprites) << 8);}while(0)

  Enable and set which backgrounds will be visible and whether sprites will too or not, for Window 1. You'll then have to configure it with PA_SetWin1XY.

- #define **PA_DisableWin1**(screen) DISPCNTL(screen) &= ∼WINDOW1

  Disable the second window...

- #define **PA_DisableWinObj**(screen) DISPCNTL(screen) &= ∼WINDOWOBJ

  Disable the object window...

- #define **PA_SetOutWin**(screen, bg_sprites) do{WINOUT(screen) &= ∼255; WINOUT(screen) |= bg_sprites;}while(0)

  Set which backgrounds will be visible and whether sprites will too or not, outside of the windows.

Functions

- static void **PA_EnableWinObj** (u8 screen, u16 bg_sprites)

  Enable and set which backgrounds will be visible and whether sprites will too or not, for Object Winodw (created from sprites in Window mode).

- static void **PA_WindowFade** (u8 screen, u8 type, u8 time)

  This allows you to do fade in and out, using the window system.

3.27.1 Detailed Description

Set up 2 windows and a possible object window...

3.27.2 Define Documentation

3.27.2.1 #define PA_SetWin1XY(screen, x1, y1, x2, y2) do{WIN1X(screen) = x2 + ((x1) << 8); WIN1Y(screen) = y2 + ((y1) << 8);}while(0)

Set the X et Y coordinates of the rectangular second window. You'll also have to use PA_SetWin1 to chose which Backgrounds are visible and if sprites are too...

Parameters:

screen Screen...

x1 X coordinate of the top left point

y1 Y coordinate of the top left point

x2 X coordinate of the bottom right point

y2 Y coordinate of the bottom right point

3.27.2.2 #define PA_EnableWin0(screen, bg_sprites) do{DISPCNTL(screen) |= WINDOW0; WININ(screen) &= 255; WININ(screen) |= (bg_sprites);}while(0)

Enable and set which backgrounds will be visible and whether sprites will too or not, for Window 0. You'll then have to configure it with PA_SetWin0XY.

Parameters:

screen Screen...

bg_sprites Backgrounds and sprites, use the following macro : WIN_BG0 | WIN_BG1 | WIN_BG2 | WIN_BG3 | WIN_OBJ | WIN_SFX (for special effects)

3.27.2.3 #define PA_DisableWin0(screen) DISPCNTL(screen) &= ~WINDOW0

Disable the first window...

Parameters:

screen Screen...

3.27.2.4 #define PA_EnableWin1(screen, bg_sprites) do{DISPCNTL(screen) |= WINDOW1; WININ(screen) &= 255; WININ(screen) |= ((bg_sprites) << 8);}while(0)

Enable and set which backgrounds will be visible and whether sprites will too or not, for Window 1. You'll then have to configure it with PA_SetWin1XY.

Parameters:

screen Screen...

bg_sprites Backgrounds and sprites, use the following macro : WIN_BG0 | WIN_BG1 | WIN_BG2 | WIN_BG3 | WIN_OBJ | WIN_SFX (for special effects)

3.27.2.5 #define PA_DisableWin1(screen) DISPCNTL(screen) &= ~WINDOW1

Disable the second window...

Parameters:

screen Screen...

3.27.2.6 #define PA_DisableWinObj(screen) DISPCNTL(screen) &= ~WINDOWOBJ

Disable the object window...

Parameters:

screen Screen...

3.27.2.7 #define PA_SetOutWin(screen, bg_sprites) do{WINOUT(screen) &= ~255; WINOUT(screen) |= bg_sprites;}while(0)

Set which backgrounds will be visible and whether sprites will too or not, outside of the windows.

Parameters:

screen Screen...

bg_sprites Backgrounds and sprites, use the following macro : WIN_BG0 | WIN_BG1 | WIN_BG2 | WIN_BG3 | WIN_OBJ

3.27.3 Function Documentation

3.27.3.1 static inline void PA_EnableWinObj (u8 *screen*, u16 *bg_sprites*) [inline, static]

Enable and set which backgrounds will be visible and whether sprites will too or not, for Object Winodw (created from sprites in Window mode).

Parameters:

screen Screen...

bg_sprites Backgrounds and sprites, use the following macro : WIN_BG0 | WIN_BG1 | WIN_BG2 | WIN_BG3 | WIN_OBJ | WIN_SFX (for special effects)

3.27.3.2 static inline void PA_WindowFade (u8 *screen*, u8 *type*, u8 *time*) [inline, static]

This allows you to do fade in and out, using the window system.

Parameters:

screen Screen...

type Type... 8 different types are available (0-7)

time Time, from 0 to 32 (included). 0 is a completely viewable screen, 32 is completely out

3.28 C++ wrappers

Namespaces

- namespace **PA**

  PAlib C++ namespace.

Functions

- void * **operator new** (size_t size)

  Lightweight new operator.

- void * **operator new[ ]** (size_t size)

  Lightweight new operator.

- void **operator delete** (void *p)

  Lightweight delete operator.

- void **operator delete[ ]** (void *p)

  Lightweight delete operator.

3.28.1 Detailed Description

C++ wrappers for PAlib

3.29 ASlib functions

Data Structures

- struct **SoundInfo**

  Sound info.

Defines

- #define **AS_SoundQuickPlay**(name) AS_SoundDefaultPlay((u8∗)name, (u32)name##_size, 127, 64, false, 0)

  Easiest way to play a sound, using default settings.

Enumerations

- enum **MP3Command** { ,

  MP3CMD_INIT = 8, **MP3CMD_STOP** = 16, **MP3CMD_PLAY** = 32, **MP3CMD_PAUSE** = 64,

  MP3CMD_SETRATE = 128 }

  MP3 commands.

- enum **SoundCommand** { ,

  SNDCMD_STOP = 2, **SNDCMD_PLAY** = 4, **SNDCMD_SETVOLUME** = 8, **SNDCMD_SETPAN** = 16,

  SNDCMD_SETRATE = 32, **SNDCMD_SETMASTERVOLUME** = 64 }

  Sound commands.

- enum **MP3Status** {

  MP3ST_STOPPED = 0, **MP3ST_PLAYING** = 1, **MP3ST_PAUSED** = 2, **MP3ST_OUT_OF_DATA** = 4,

  MP3ST_DECODE_ERROR = 8, **MP3ST_INITFAILED** = 16 }

  MP3 states.

- enum **AS_MODE** { **AS_MODE_MP3** = 1, **AS_MODE_SURROUND** = 2, **AS_MODE_16CH** = 4, **AS_MODE_8CH** = 8 }

  ASlib modes.

- enum **AS_DELAY** { **AS_NO_DELAY** = 0, **AS_SURROUND** = 1, **AS_-REVERB** = 4 }

  Delay values.

- enum **AS_SOUNDFORMAT** { **AS_PCM_8BIT** = 0, **AS_PCM_16BIT** = 1, **AS_ADPCM** = 2 }

Sound formats.

Functions

- void **AS_Init** (u8 mode)

  Initialize ASlib.

- static void **AS_ReserveChannel** (u8 channel)

  Reserve a particular DS channel (so it won't be used for the sound pool).

- static void **AS_SetMasterVolume** (u8 volume)

  Set the master volume (0..127).

- static void **AS_SetDefaultSettings** (u8 format, s32 rate, u8 delay)

  Set the default sound settings.

- int **AS_SoundPlay** (**SoundInfo** sound)

  Play a sound using the priority system. Returns the sound channel allocated or -1 if the sound was skipped.

- static int **AS_SoundDefaultPlay** (u8 ∗data, u32 size, u8 volume, u8 pan, u8 loop, u8 prio)

  Play a sound using the priority system with the default settings. Returns the sound channel allocated or -1 if the sound was skipped.

- void **AS_SetSoundPan** (u8 chan, u8 pan)

  Set the panning of a sound (0=left, 64=center, 127=right).

- void **AS_SetSoundVolume** (u8 chan, u8 volume)

  Set the volume of a sound (0..127).

- void **AS_SetSoundRate** (u8 chan, u32 rate)

  Set the sound sample rate.

- static void **AS_SoundStop** (u8 chan)

  Stop playing a sound.

- void **AS_SoundDirectPlay** (u8 chan, **SoundInfo** sound)

  Play a sound directly using the given channel.

- void **AS_MP3DirectPlay** (u8 ∗buffer, u32 size)

  Play a MP3 directly from memory.

- void **AS_MP3StreamPlay** (const char ∗path)

  Play a MP3 stream.

- static void **AS_MP3Pause** ()

  Pause a MP3.

- static void **AS_MP3Unpause** ()

  Unpause a MP3.

- static void **AS_MP3Stop** ()

  Stop a MP3.

- static int **AS_GetMP3Status** ()

  Get the current MP3 status.

- static void **AS_SetMP3Volume** (u8 volume)

  Set the MP3 volume (0..127).

- void **AS_SetMP3Pan** (u8 pan)

  Set the MP3 panning (0=left, 64=center, 127=right).

- static void **AS_SetMP3Delay** (u8 delay)

  Set the default MP3 delay mode (warning: high values can cause glitches).

- static void **AS_SetMP3Loop** (u8 loop)

  Set the MP3 loop mode (false = one shot, true = loop indefinitely).

- static void **AS_SetMP3Rate** (s32 rate)

  Set the MP3 sample rate.

- void **AS_SoundVBL** ()

  Regenerate buffers for MP3 stream. Must be called each VBlank (only needed if mp3 is used).

3.29.1 Detailed Description

Functions to play RAW sounds and ∗shrug∗ MP3s.

3.29.2 Enumeration Type Documentation

3.29.2.1 enum MP3Command

MP3 commands.

Enumerator:

MP3CMD_INIT Initialize.

MP3CMD_STOP Stop.

MP3CMD_PLAY Play.

MP3CMD_PAUSE Pause.

MP3CMD_SETRATE Set rate.

3.29.2.2 enum SoundCommand

Sound commands.

Enumerator:

SNDCMD_STOP Stop.

SNDCMD_PLAY Play.

SNDCMD_SETVOLUME Set volume.

SNDCMD_SETPAN Set pan.

SNDCMD_SETRATE Set rate.

SNDCMD_SETMASTERVOLUME Set master volume.

3.29.2.3 enum MP3Status

MP3 states.

Enumerator:

MP3ST_STOPPED Stopped.

MP3ST_PLAYING Playing.

MP3ST_PAUSED Paused.

MP3ST_OUT_OF_DATA Out of data.

MP3ST_DECODE_ERROR Decoding error.

MP3ST_INITFAILED Initialization failed.

3.29.2.4 enum AS_MODE

ASlib modes.

Enumerator:

AS_MODE_MP3 use mp3

AS_MODE_SURROUND use surround

AS_MODE_16CH use all DS channels

AS_MODE_8CH use DS channels 1-8 only

3.29.2.5 enum AS_DELAY

Delay values.

Enumerator:

AS_NO_DELAY 0 ms delay

AS_SURROUND 16 ms delay

AS_REVERB 66 ms delay

3.29.2.6 enum AS_SOUNDFORMAT

Sound formats.

Enumerator:

AS_PCM_8BIT 8-bit PCM

AS_PCM_16BIT 16-bit PCM

AS_ADPCM 4-bit ADPCM

Chapter 4

Namespace Documentation

4.1 PA Namespace Reference

PAlib C++ namespace.

Data Structures

- class **Application**
  Simple application abstraction layer for PAlib C++ programs.
- class **Fixed**
  Fixed-point wrapper class.
- class **Point**
  Fixed-point point class.
- class **Sprite**
  Wrapper class for sprites.
- class **HandleProvider**
  Handle provider, use it to get dynamic sprite numbers for example.

4.1.1 Detailed Description

PAlib C++ namespace.

Chapter 5

Data Structure Documentation

5.1 PA::Application Class Reference

Simple application abstraction layer for PAlib C++ programs.

Public Member Functions

- void **run** ()
  Runs the application.

Protected Member Functions

- virtual void **init** ()
  Initialization function.
- virtual bool **update** ()
  Update function.
- virtual void **render** ()
  Render function.
- virtual void **cleanup** ()
  Cleanup function (optional).

5.1.1 Detailed Description

Simple application abstraction layer for PAlib C++ programs.

5.2 PA::Fixed Class Reference

Fixed-point wrapper class.

Public Member Functions

- **Fixed** ()
  Empty constructor.
- **Fixed** (const **Fixed** &a)
  Copy constructor.
- **Fixed** (int a)
  int constructor.
- **Fixed** (float a)
  float constructor.
- **operator int** () const
  int cast.
- **operator float** () const
  float cast.
- **operator bool** () const
  bool cast.
- **operator char** () const
  char cast.
- **operator short** () const
  short cast.
- **operator long long** () const
  long long cast.
- **Fixed** & **operator=** (const **Fixed** &a)
  Assignment operator.
- **Fixed operator+** (const **Fixed** &a) const
  Addition operator. int and float versions also available.
- **Fixed operator-** (const **Fixed** &a) const
  Subtraction operator. int and float versions also available.
- **Fixed operator**∗ (const **Fixed** &a) const

Multiplication operator. int and float versions also available.

- **Fixed operator/** (const **Fixed** &a) const

  Division operator. int and float versions also available.

- **Fixed operator%** (const **Fixed** &a) const

  Modulo operator. int and float versions also available.

- **Fixed operator++** ()

  Pre-increment operator.

- **Fixed operator--** ()

  Pre-decrement operator.

- **Fixed operator++** (int)

  Post-increment operator.

- **Fixed operator--** (int)

  Post-decrement operator.

- **Fixed operator-** () const

  Negation operator.

- **Fixed operator$\sim$** () const

  Binary negation operator.

- **Fixed** & **operator+=** (const **Fixed** &a)

  Addition and assignment operator. int and float versions also available.

- **Fixed** & **operator-=** (const **Fixed** &a)

  Subtraction and assignment operator. int and float versions also available.

- **Fixed** & **operator$*$=** (const **Fixed** &a)

  Multiplication and assignment operator. int and float versions also available.

- **Fixed** & **operator%=** (const **Fixed** &a)

  Modulo and assignment operator. int and float versions also available.

- bool **operator==** (const **Fixed** &a) const

  Equals operator. int and float versions also available.

- bool **operator!=** (const **Fixed** &a) const

  Not-equals operator. int and float versions also available.

- bool **operator$<$=** (const **Fixed** &a) const

  Less-or-equal operator. int and float versions also available.

- bool **operator>=** (const **Fixed** &a) const

  Greater-or-equal operator. int and float versions also available.

- bool **operator<** (const **Fixed** &a) const

  Less-than operator. int and float versions also available.

- bool **operator>** (const **Fixed** &a) const

  Greater-than operator. int and float versions also available.

- **Fixed operator<<** (int a) const

  Left shift operator.

- **Fixed operator>>** (int a) const

  Right shift operator.

- **Fixed** & **operator<<=** (int a)

  Left shift and assign operator.

- **Fixed** & **operator>>=** (int a)

  Right shift and assign operator.

- **Fixed operator&** (u32 a) const

  Binary AND operator, u32 version.

- **Fixed operator|** (u32 a) const

  Binary OR operator, u32 version.

- **Fixed operator^** (u32 a) const

  Binary XOR operator, u32 version.

- **Fixed operator&** (const **Fixed** &a) const

  *Binary AND operator, **Fixed** (p. 180) version.*

- **Fixed operator|** (const **Fixed** &a) const

  *Binary OR operator, **Fixed** (p. 180) version.*

- **Fixed operator^** (const **Fixed** &a) const

  *Binary XOR operator, **Fixed** (p. 180) version.*

- **Fixed** & **operator&=** (u32 a)

  Binary AND assignment, u32 version.

- **Fixed** & **operator|=** (u32 a)

  Binary OR assignment, u32 version.

- **Fixed** & **operator^=** (u32 a)

  Binary XOR assignment, u32 version.

- **Fixed** & **operator&=** (const **Fixed** &a)

  *Binary AND assignment, **Fixed** (p. 180) version.*

- **Fixed** & **operator|=** (const **Fixed** &a)

  *Binary OR assignment, **Fixed** (p. 180) version.*

- **Fixed** & **operator^=** (const **Fixed** &a)

  *Binary XOR assignment, **Fixed** (p. 180) version.*

- **Fixed sqrt** () const

  Gets the square root.

- **Fixed abs** () const

  Gets the absolute value.

- int **raw** () const

  Gets the raw Q12 fixed point number.

Static Public Member Functions

- static **Fixed r2f** (int a)

  *Creates a **Fixed** (p. 180) object using a raw Q12 fixed point number.*

5.2.1 Detailed Description

Fixed-point wrapper class.

5.3 PA::HandleProvider< NHANDLES > Class Template Reference

Handle provider, use it to get dynamic sprite numbers for example.

Public Member Functions

- **HandleProvider** ()
  Constructor.
- int **newhandle** ()
  Get a new handle.
- void **deletehandle** (int handle)
  Delete a handle.

5.3.1 Detailed Description

template<int NHANDLES> class PA::HandleProvider< NHANDLES >

Handle provider, use it to get dynamic sprite numbers for example.

5.4 PA_BgStruct Struct Reference

Background structure.

Data Fields

- int **BgType**
  Type of background.
- int **width**
  Width of background in pixels.
- int **height**
  Height of background in pixels.
- const void * **BgTiles**
  Pointer to background tiles.
- const void * **BgMap**
  Pointer to background map.
- size_t **BgTiles_size**
  Size of tiles in bytes.
- const void * **BgPalette**
  Pointer to palette.
- const void * **FontSizes**
  Pointer to font sizes.
- size_t **BgMap_size**
  Size of the map in bytes.
- int **FontHeight**
  Height of the font in pixels.

5.4.1 Detailed Description

Background structure.

5.5 PA_FifoMsg Struct Reference

Represents a message sent through Fifo.

Data Fields

- u32 **type**

  Type of message.

- union {
  struct {
  u16 **tdiode1**
  TSC temperature diode 1.
  u16 **tdiode2**
  TSC temperature diode 1.
  u32 **temperature**
  TSC computed temperature.
  u16 **battery**
  TSC battery.
  u8 **micvol**
  Microphone volume.
  u8 **extra**
  Extra byte - used as padding for now.
  } **InputMsg**
  Input message data.
  struct {
  u8 * **buffer**
  Buffer to record microphone data.
  u32 **length**
  Length of the buffer in bytes.
  } **MicMsg**
  Microphone record message data.
  struct {
  u8 **brightness**
  Brightness of the lights (0-3).
  } **DSLBrightMsg**
  DS lite brightness message data.
  struct {
  u32 **freq**
  Frequency (in hertz).
  u8 **chan**
  Channel.
  u8 **vol**
  Volume (0-127).
  u8 **pan**
  Pan (0-64-127).
  u8 **duty**
  Duty (0-7).

} **PSGMsg**
PSG play message data.
};

--

5.5.1 Detailed Description

Represents a message sent through Fifo.

5.6 PA_Point Struct Reference

Simple point structure.

Data Fields

- int **x**
  X value.
- int **y**
  Y value.

5.6.1 Detailed Description

Simple point structure.

5.7 PA_TransferRegion Struct Reference

PAlib transfer region type.

Data Fields

- vuint16 **tdiode1**
  TSC temperature diode 1.
- vuint16 **tdiode2**
  TSC temperature diode 2.
- vuint32 **temperature**
  TSC computed temperature.
- vuint16 **battery**
  TSC battery.
- vuint8 **micvol**
  Microphone volume.
- vuint8 **extra**
  Extra field - used as padding for now.
- LEGACY vuint32 **mailData**
  Legacy IPC field.

5.7.1 Detailed Description

PAlib transfer region type.

5.8 PA::Point Class Reference

Fixed-point point class.

Public Member Functions

- **operator PA_Point** () const

  *Convert the object to a **PA_Point*** (p. *188*) *structure.*

Data Fields

- **Fixed x**

  X value.

- **Fixed y**

  Y value.

5.8.1 Detailed Description

Fixed-point point class.

5.9 SoundInfo Struct Reference

Sound info.

Data Fields

- u8 * **data**
  Pointer to data.
- u32 **size**
  Size in bytes.
- u8 **format**
  Format (see AS_SOUNDFORMAT).
- s32 **rate**
  Rate in Hz.
- u8 **volume**
  Volume (0-127).
- s8 **pan**
  Pan (0-64-127).
- u8 **loop**
  Loop (0 or 1).
- u8 **priority**
  Priority.
- u8 **delay**
  Delay.

5.9.1 Detailed Description

Sound info.

5.10 PA::Sprite Class Reference

Wrapper class for sprites.

Public Member Functions

- **Sprite** ()
  Empty constructor.
- **Sprite** (int scr, int sprn)
  Normal constructor.
- void **init** (int scr, int sprn)
  Initialize function.
- void **create** (void *gfx, int shape, int size, int paln)
  Create sprite.
- void **create** (u16 gfx, int shape, int size, int paln)
  Create sprite from existing GFX.
- void **remove** ()
  Delete sprite.
- void **setpalette** (int paln)
  Set palette.
- void **setgfx** (int gfxn)
  Set GFX.
- void **render** ()
  Render (more like update position).
- void **move** (const **Fixed** &x, const **Fixed** &y)
  Move (fixed point version).
- void **move** (int x, int y)
  Move (integer version).
- void **hflip** (bool flip)
  Set HFlip.
- void **vflip** (bool flip)
  Set VFlip.
- void **dblsize** (bool dblsize)

Set doublesize.

- void **priority** (int prio)

  Set priority.

- void **bindrotset** (int rotset)

  Bind rotset.

- void **debindrotset** ()

  Debind rotset.

- void **rotate** (int angle)

  Rotate.

- void **zoom** (int zx, int zy)

  Zoom.

- void **rotozoom** (int angle, int zx, int zy)

  Rotate and zoom.

- void **frame** (int frame)

  Set frame.

- void **startanim** (int begin, int end, int speed, int animtype=ANIM_LOOP, int ncycles=-1)

  Start animation.

- void **pauseanim** (bool pause=true)

  Pause animation.

- void **stopanim** ()

  Stop animation.

- void **animspeed** (int speed)

  Set animation speed.

5.10.1 Detailed Description

Wrapper class for sprites.

Chapter 6

Example Documentation

6.1 Backgrounds/Effects/Mode7/source/main.c

```
// Mode 7 example.

// Includes
#include <PA9.h>

#include "all_gfx.h"

int main(){
    PA_Init();

    PA_SetVideoMode(0, 2);  //screen, mode
    PA_SetVideoMode(1, 2);  //screen, mode

    // Yup, we use the standard bg load function!
    PA_LoadBackground(0, //screen
                      3, // background number
                      &Rot); // background name in PAGfx
    PA_LoadBackground(1, 3, &Rot);

    // Wraparound (!)
    PA_SetBgWrap(0, 3, 1);
    PA_SetBgWrap(1, 3, 1);

    PA_LoadDefaultText(1, 0);

    PA_InitMode7(3);

    u16 angle = 0;
    u16 height = 8192;

    while(true){
        // Change the angle
        angle += Pad.Held.Right - Pad.Held.Left;
        angle &= 511;
        PA_Mode7Angle(angle);

        // Move left/right
        PA_Mode7MoveLeftRight(Pad.Held.A - Pad.Held.Y);
```

```
        // Move Forward/backward
        PA_Mode7MoveForwardBack(Pad.Held.Up - Pad.Held.Down);

        // Height
        height += (Pad.Held.X - Pad.Held.B)<<7;
        PA_Mode7Height(height);

        PA_OutputText(1, 0, 0, "Angle  : %d    ", angle);
        PA_OutputText(1, 0, 1, "Height : %d    ", height);

        PA_WaitForVBL();
    }
}
```

6.2 Text/Normal/HelloWorld/source/main.c

```
// Hello World Program //

// Lines starting with two slashes are ignored by the compiler
// Basically you can use them to comment what are you doing
// In fact, this kind of lines are called comments :P

// Include PAlib so that you can use it
#include <PA9.h>

int main(){
    // Initialize PAlib
    PA_Init();

    // Load the default text font
    PA_LoadDefaultText(1, // Top screen
                       2); // Background #2

    // Write the text "Hello World"
    PA_OutputSimpleText(1, // Top screen
                        1, // X position 1*8 = 8
                        1, // Y position 1*8 = 8
                        "Hello World");

    // Infinite loop to keep the program running
    while(true){
        // Wait until the next frame.
        // The DS runs at 60 frames per second.
        PA_WaitForVBL();
    }
}
```

Index

16color pseudo-bitmap mode, 7
3D Sprite System, 13

AS_ADPCM
    ASlib, 175
AS_DELAY
    ASlib, 174
AS_MODE
    ASlib, 174
AS_MODE_16CH
    ASlib, 174
AS_MODE_8CH
    ASlib, 174
AS_MODE_MP3
    ASlib, 174
AS_MODE_SURROUND
    ASlib, 174
AS_NO_DELAY
    ASlib, 175
AS_PCM_16BIT
    ASlib, 175
AS_PCM_8BIT
    ASlib, 175
AS_REVERB
    ASlib, 175
AS_SOUNDFORMAT
    ASlib, 175
AS_SURROUND
    ASlib, 175
ASlib
    AS_ADPCM, 175
    AS_DELAY, 174
    AS_MODE, 174
    AS_MODE_16CH, 174
    AS_MODE_8CH, 174
    AS_MODE_MP3, 174
    AS_MODE_SURROUND, 174
    AS_NO_DELAY, 175
    AS_PCM_16BIT, 175
    AS_PCM_8BIT, 175
    AS_REVERB, 175
    AS_SOUNDFORMAT, 175
    AS_SURROUND, 175
    MP3CMD_INIT, 173
    MP3CMD_PAUSE, 174
    MP3CMD_PLAY, 174
    MP3CMD_SETRATE, 174
    MP3CMD_STOP, 173
    MP3Command, 173
    MP3ST_DECODE_ERROR, 174
    MP3ST_INITFAILED, 174
    MP3ST_OUT_OF_DATA, 174
    MP3ST_PAUSED, 174
    MP3ST_PLAYING, 174
    MP3ST_STOPPED, 174
    MP3Status, 174
    SNDCMD_PLAY, 174
    SNDCMD_-
        SETMASTERVOLUME,
        174
    SNDCMD_SETPAN, 174
    SNDCMD_SETRATE, 174
    SNDCMD_SETVOLUME, 174
    SNDCMD_STOP, 174
    SoundCommand, 174
ASlib functions, 171

Background Transition Effects, 44
Bg Modes on 2 Screens, 153
BgLargeMap
    PA_InfLargeScrollX, 21
    PA_InfLargeScrollXY, 22
    PA_InfLargeScrollY, 22
    PA_LargeScrollX, 22
    PA_LargeScrollXY, 23
    PA_LargeScrollY, 22
    PA_LoadLargeBg, 20
    PA_LoadLargeBgEx, 21
    PA_LoadPAGfxLargeBg, 20
BgRot
    PA_LoadPAGfxRotBg, 25
    PA_LoadRotBg, 24

PA_SetBgRot, 25
BgTiles
PA_BgInvalid, 35
PA_BgLarge, 35
PA_BgNormal, 35
PA_BgRot, 35
PA_BGScrollX, 37
PA_BGScrollY, 37
PA_BgUnlimited, 35
PA_ClearBg, 40
PA_DeleteBg, 36
PA_DeleteMap, 36
PA_DeleteTiles, 36
PA_EasyBgGetPixel, 40
PA_EasyBgGetPixelCol, 41
PA_EasyBgLoad, 33
PA_EasyBgLoadPtr, 34
PA_EasyBgScrollX, 40
PA_EasyBgScrollXY, 40
PA_EasyBgScrollY, 40
PA_Font1bit, 35
PA_Font4bit, 35
PA_Font8bit, 35
PA_HideBg, 30
PA_InitBg, 35
PA_InitParallaxX, 41
PA_InitParallaxY, 42
PA_LoadBackground, 37
PA_LoadBg, 32
PA_LoadBgMap, 36
PA_LoadBgTiles, 31
PA_LoadBgTilesEx, 35
PA_LoadSimpleBg, 32
PA_LoadTiledBg, 31
PA_ParallaxScrollX, 42
PA_ParallaxScrollXY, 42
PA_ParallaxScrollY, 42
PA_ReLoadBgTiles, 36
PA_ResetBg, 31
PA_ResetBgSysScreen, 35
PA_SetBgPrio, 39
PA_SetBgPrioSeq, 39
PA_SetBgWrap, 41
PA_SetLargeMapTile, 38
PA_SetMapTile, 38
PA_SetMapTileAll, 33
PA_SetMapTileHflip, 38
PA_SetMapTilePal, 39
PA_SetMapTileVflip, 38
PA_ShowBg, 30
bgtrans
PA_BgTransCenter, 45
PA_BgTransDiag, 45
PA_BgTransLeftRight, 45
PA_BgTransUpDown, 44
PA_InitBgTrans, 44
PA_InitBgTransEx, 44
Bitmap
PA_16bitDraw, 57
PA_8bitDraw, 57
PA_Clear16bitBg, 52
PA_Clear8bitBg, 52
PA_Draw16bitLine, 55
PA_Draw16bitLineEx, 55
PA_Draw16bitRect, 56
PA_Draw8bitLine, 55
PA_Draw8bitLineEx, 56
PA_Get16bitPixel, 51
PA_Get8bitPixel, 54
PA_GetBmpHeight, 58
PA_GetBmpWidth, 58
PA_Init16bitBg, 53
PA_Init8bitBg, 52
PA_InitBig8bitBg, 53
PA_Load16bitBitmap, 52
PA_Load8bitBitmap, 51
PA_LoadBmp, 58
PA_LoadBmpEx, 58
PA_LoadBmpToBuffer, 57
PA_LoadJpeg, 57
PA_Put16bitPixel, 54
PA_Put2_8bitPixels, 53
PA_Put4_8bitPixels, 54
PA_Put8bitPixel, 53
PA_PutDouble8bitPixels, 54
PA_SetDrawSize, 51
Bitmap mode, 49

C++ wrappers, 170
c16
PA_16c8X4, 10
PA_16c8X6, 11
PA_16c8X8, 11
PA_16c8Xi, 11
PA_16cClearZone, 12
PA_16cCustomFont, 8
PA_16cErase, 9
PA_16cGetPixel, 12
PA_16cPutPixel, 10
PA_16cText, 9

PA_Add16cFont, 10
PA_Init16cBg, 9
PA_InitComplete16c, 9

Debug
PA_Assert, 47
PA_iDeaS_DebugOutput, 47
PA_iDeaS_DebugPrintf, 48
Debugging utilities, 47
DS Motion functions, 91

f3DSprites
PA_3DGetSpriteAnimFrame, 17
PA_3DGetSpriteAnimSpeed, 18
PA_3DGetSpriteNCycles, 18
PA_3DSetSpriteAnimFrame, 17
PA_3DSetSpriteAnimSpeed, 17
PA_3DSetSpriteNCycles, 18
PA_3DSpriteAnimPause, 18
PA_3DStartSpriteAnim, 16
PA_3DStartSpriteAnimEx, 16
PA_3DStopSpriteAnim, 17
Fake 16bit bitmap mode, 60
Fake16bit
PA_ClearFake16bitBg, 61
PA_DrawFake16bitLine, 63
PA_DrawFake16bitRect, 62
PA_Fake16bitLoadBmp, 62
PA_Fake16bitLoadBmpEx, 62
PA_Fake16bitLoadGif, 63
PA_Fake16bitLoadJpeg, 63
PA_GetFake16bitPixel, 61
PA_InitFake16bitBg, 63
PA_LoadFake16bitBitmap, 61
PA_PutFake16bitPixel, 61

General
PA_CloseLidSound, 68
PA_CloseLidSound2, 68
PA_LegacyIPCInit, 67
PA_Locate, 70
PA_MSG_DSLBRIGHT, 69
PA_MSG_INPUT, 69
PA_MSG_MIC, 69
PA_MSG_MICSTOP, 69
PA_MSG_PSG, 69
PA_SetAutoCheckLid, 69
PA_SetDSLBrightness, 70
PA_SetLedBlink, 69
PA_SetScreenLight, 70
PA_SetVideoMode, 69
PA_WaitFor, 68
General Functions, 65
Gif
PA_GetGifHeight, 72
PA_GetGifWidth, 71
PA_GifAnimSpeed, 72
PA_GifAnimStop, 72
PA_GifSetEndFrame, 73
PA_GifSetStartFrame, 72
PA_LoadGif, 72
PA_LoadGifXY, 72
Gif functions, 71

Key input system, 78
Keyboard, 74
PA_InitCustomKeyboard, 75
PA_KeyboardIn, 76
PA_LoadDefaultKeyboard, 75
PA_LoadKeyboard, 75
PA_ScrollKeyboardX, 76
PA_ScrollKeyboardXY, 76
PA_ScrollKeyboardY, 76
PA_SetKeyboardColor, 76
PA_SetKeyboardScreen, 77
Keys
PA_MoveSprite, 79
PA_MoveSpriteDistance, 80
PA_MoveSpriteEx, 80
PA_MoveSpritePix, 80
PA_SpriteStylusOver, 81
PA_SpriteStylusOverEx, 80
PA_SpriteTouched, 81
PA_SpriteTouchedEx, 81
PA_StylusInZone, 79

Math
PA_AdjustAngle, 85
PA_Distance, 85
PA_divf32, 86
PA_GetAngle, 85
PA_modf32, 86
PA_mulf32, 86
PA_RandMax, 84
PA_RandMinMax, 84
PA_sqrtf32, 86
PA_SRand, 84
PA_TrueDistance, 85
Math functions, 83
Micro

PA_MicReplay, 87
PA_MicStartRecording, 87
Microphone, 87
Mode 7 commands, 88
Mode7
PA_InitMode7, 88
PA_Mode7Angle, 89
PA_Mode7Height, 90
PA_Mode7MoveForwardBack, 89
PA_Mode7MoveLeftRight, 89
PA_Mode7SetPointXZ, 90
PA_Mode7X, 89
PA_Mode7Z, 89
MP3CMD_INIT
ASlib, 173
MP3CMD_PAUSE
ASlib, 174
MP3CMD_PLAY
ASlib, 174
MP3CMD_SETRATE
ASlib, 174
MP3CMD_STOP
ASlib, 173
MP3Command
ASlib, 173
MP3ST_DECODE_ERROR
ASlib, 174
MP3ST_INITFAILED
ASlib, 174
MP3ST_OUT_OF_DATA
ASlib, 174
MP3ST_PAUSED
ASlib, 174
MP3ST_PLAYING
ASlib, 174
MP3ST_STOPPED
ASlib, 174
MP3Status
ASlib, 174

Old large background system, 19

PA, 177
PA::Application, 179
PA::Fixed, 180
PA::HandleProvider, 184
PA::Point, 190
PA::Sprite, 192
PA_16bitDraw
Bitmap, 57
PA_16c8X4
c16, 10
PA_16c8X6
c16, 11
PA_16c8X8
c16, 11
PA_16c8Xi
c16, 11
PA_16cClearZone
c16, 12
PA_16cCustomFont
c16, 8
PA_16cErase
c16, 9
PA_16cGetPixel
c16, 12
PA_16cPutPixel
c16, 10
PA_16cText
c16, 9
PA_3DGetSpriteAnimFrame
f3DSprites, 17
PA_3DGetSpriteAnimSpeed
f3DSprites, 18
PA_3DGetSpriteNCycles
f3DSprites, 18
PA_3DSetSpriteAnimFrame
f3DSprites, 17
PA_3DSetSpriteAnimSpeed
f3DSprites, 17
PA_3DSetSpriteNCycles
f3DSprites, 18
PA_3DSetSpritePalCol
Palette, 97
PA_3DSpriteAnimPause
f3DSprites, 18
PA_3DStartSpriteAnim
f3DSprites, 16
PA_3DStartSpriteAnimEx
f3DSprites, 16
PA_3DStopSpriteAnim
f3DSprites, 17
PA_8bitCustomFont
Text, 146
PA_8bitDraw
Bitmap, 57
PA_8bitText
Text, 149
PA_Add16cFont
c16, 10

PA_AddBitmapFont
Text, 150
PA_AdjustAngle
Math, 85
PA_Assert
Debug, 47
PA_BgInvalid
BgTiles, 35
PA_BgLarge
BgTiles, 35
PA_BgNormal
BgTiles, 35
PA_BgRot
BgTiles, 35
PA_BGScrollX
BgTiles, 37
PA_BGScrollY
BgTiles, 37
PA_BgStruct, 185
PA_BgTransCenter
bgtrans, 45
PA_BgTransDiag
bgtrans, 45
PA_BgTransLeftRight
bgtrans, 45
PA_BgTransUpDown
bgtrans, 44
PA_BgUnlimited
BgTiles, 35
PA_BoxText
Text, 148
PA_BoxTextNoWrap
Text, 149
PA_CenterSmartText
Text, 150
PA_Clear16bitBg
Bitmap, 52
PA_Clear8bitBg
Bitmap, 52
PA_ClearBg
BgTiles, 40
PA_ClearFake16bitBg
Fake16bit, 61
PA_ClearTextBg
Text, 151
PA_CloneSprite
Sprite, 117
PA_CloseLidSound
General, 68
PA_CloseLidSound2
General, 68
PA_Create16bitSprite
Sprite, 120
PA_Create16bitSpriteEx
Sprite, 119
PA_Create16bitSpriteFromGfx
Sprite, 120
PA_CreateGfx
Sprite, 118
PA_CreateSprite
Sprite, 118
PA_CreateSpriteEx
Sprite, 118
PA_CreateSpriteExFromGfx
Sprite, 121
PA_CreateSpriteFromGfx
Sprite, 120
PA_DeleteBg
BgTiles, 36
PA_DeleteGfx
Sprite, 122
PA_DeleteMap
BgTiles, 36
PA_DeleteSprite
Sprite, 122
PA_DeleteTiles
BgTiles, 36
PA_DisableBgMosaic
SpecialFx, 103
PA_DisableSpecialFx
SpecialFx, 104
PA_DisableWin0
Window, 167
PA_DisableWin1
Window, 168
PA_DisableWinObj
Window, 168
PA_Distance
Math, 85
PA_divf32
Math, 86
PA_Draw16bitLine
Bitmap, 55
PA_Draw16bitLineEx
Bitmap, 55
PA_Draw16bitRect
Bitmap, 56
PA_Draw8bitLine
Bitmap, 55
PA_Draw8bitLineEx

Bitmap, 56
PA_DrawFake16bitLine
Fake16bit, 63
PA_DrawFake16bitRect
Fake16bit, 62
PA_DualBGScrollX
TileDual, 160
PA_DualBGScrollXY
TileDual, 161
PA_DualBGScrollY
TileDual, 161
PA_DualCloneSprite
SpriteDual, 141
PA_DualCreate16bitSprite
SpriteDual, 135
PA_DualCreate16bitSpriteEx
SpriteDual, 135
PA_DualCreateSprite
SpriteDual, 133
PA_DualCreateSpriteEx
SpriteDual, 134
PA_DualCreateSpriteExFromGfx
SpriteDual, 136
PA_DualCreateSpriteFromGfx
SpriteDual, 136
PA_DualDeleteBg
TileDual, 160
PA_DualDeleteSprite
SpriteDual, 137
PA_DualEasyBgLoad
TileDual, 159
PA_DualEasyBgScrollX
TileDual, 161
PA_DualEasyBgScrollXY
TileDual, 162
PA_DualEasyBgScrollY
TileDual, 161
PA_DualGetSpriteAnimFrame
SpriteDual, 143
PA_DualGetSpriteAnimSpeed
SpriteDual, 143
PA_DualHideBg
TileDual, 160
PA_DualInfLargeScrollX
TileDual, 162
PA_DualInfLargeScrollXY
TileDual, 162
PA_DualInfLargeScrollY
TileDual, 162
PA_DualInitParallaxX
TileDual, 163
PA_DualInitParallaxY
TileDual, 164
PA_DualLargeScrollX
TileDual, 163
PA_DualLargeScrollXY
TileDual, 163
PA_DualLargeScrollY
TileDual, 163
PA_DualLoadBackground
TileDual, 162
PA_DualLoadBg
TileDual, 157
PA_DualLoadBgPal
PaletteDual, 100
PA_DualLoadLargeBg
TileDual, 158
PA_DualLoadLargeBgEx
TileDual, 159
PA_DualLoadPAGfxLargeBg
TileDual, 158
PA_DualLoadPal
PaletteDual, 98
PA_DualLoadPal16
PaletteDual, 99
PA_DualLoadRotBg
TileDual, 156
PA_DualLoadSimpleBg
TileDual, 156
PA_DualLoadSpritePal
PaletteDual, 99
PA_DualLoadTiledBg
TileDual, 155
PA_DualParallaxScrollX
TileDual, 164
PA_DualParallaxScrollXY
TileDual, 165
PA_DualParallaxScrollY
TileDual, 164
PA_DualSetBgColor
PaletteDual, 100
PA_DualSetBgPrio
TileDual, 165
PA_DualSetPal16Neg
PaletteDual, 99
PA_DualSetPalNeg
PaletteDual, 99
PA_DualSetRotset
SpriteDual, 138
PA_DualSetRotsetNoAngle

SpriteDual, 138
PA_DualSetRotsetNoZoom
SpriteDual, 138
PA_DualSetSpriteAnim
SpriteDual, 141
PA_DualSetSpriteAnimEx
SpriteDual, 141
PA_DualSetSpriteAnimFrame
SpriteDual, 142
PA_DualSetSpriteAnimSpeed
SpriteDual, 143
PA_DualSetSpriteColors
SpriteDual, 139
PA_DualSetSpriteDblsize
SpriteDual, 139
PA_DualSetSpriteGfx
SpriteDual, 140
PA_DualSetSpriteHflip
SpriteDual, 140
PA_DualSetSpriteMode
SpriteDual, 139
PA_DualSetSpriteMosaic
SpriteDual, 140
PA_DualSetSpritePal
SpriteDual, 139
PA_DualSetSpritePrio
SpriteDual, 140
PA_DualSetSpriteRotDisable
SpriteDual, 138
PA_DualSetSpriteRotEnable
SpriteDual, 137
PA_DualSetSpriteVflip
SpriteDual, 140
PA_DualSetSpriteX
SpriteDual, 133
PA_DualSetSpriteXY
SpriteDual, 133
PA_DualSetSpriteY
SpriteDual, 133
PA_DualShowBg
TileDual, 160
PA_DualSpriteAnimPause
SpriteDual, 143
PA_DualStartSpriteAnim
SpriteDual, 142
PA_DualStartSpriteAnimEx
SpriteDual, 141
PA_DualStopSpriteAnim
SpriteDual, 142
PA_DualUpdateGfx
SpriteDual, 137
PA_DualUpdateSpriteGfx
SpriteDual, 137
PA_EasyBgGetPixel
BgTiles, 40
PA_EasyBgGetPixelCol
BgTiles, 41
PA_EasyBgLoad
BgTiles, 33
PA_EasyBgLoadPtr
BgTiles, 34
PA_EasyBgScrollX
BgTiles, 40
PA_EasyBgScrollXY
BgTiles, 40
PA_EasyBgScrollY
BgTiles, 40
PA_EnableBgMosaic
SpecialFx, 102
PA_EnableSpecialFx
SpecialFx, 103
PA_EnableWin0
Window, 167
PA_EnableWin1
Window, 167
PA_EnableWinObj
Window, 169
PA_EraseTextBox
Text, 151
PA_Fake16bitLoadBmp
Fake16bit, 62
PA_Fake16bitLoadBmpEx
Fake16bit, 62
PA_Fake16bitLoadGif
Fake16bit, 63
PA_Fake16bitLoadJpeg
Fake16bit, 63
PA_FifoMsg, 186
PA_Font1bit
BgTiles, 35
PA_Font4bit
BgTiles, 35
PA_Font8bit
BgTiles, 35
PA_Get16bitPixel
Bitmap, 51
PA_Get8bitPixel
Bitmap, 54
PA_GetAngle
Math, 85

PA_GetBmpHeight
Bitmap, 58
PA_GetBmpWidth
Bitmap, 58
PA_GetFake16bitPixel
Fake16bit, 61
PA_GetGifHeight
Gif, 72
PA_GetGifWidth
Gif, 71
PA_GetSprite16cPixel
Sprite, 128
PA_GetSpriteAnimFrame
Sprite, 126
PA_GetSpriteAnimSpeed
Sprite, 126
PA_GetSpriteColors
Sprite, 113
PA_GetSpriteDblsize
Sprite, 113
PA_GetSpriteGfx
Sprite, 116
PA_GetSpriteHflip
Sprite, 115
PA_GetSpriteLx
Sprite, 117
PA_GetSpriteLy
Sprite, 117
PA_GetSpriteMode
Sprite, 114
PA_GetSpriteMosaic
Sprite, 114
PA_GetSpriteNCycles
Sprite, 127
PA_GetSpritePal
Sprite, 112
PA_GetSpritePixel
Sprite, 128
PA_GetSpritePrio
Sprite, 116
PA_GetSpriteVflip
Sprite, 115
PA_GetSpriteX
Sprite, 112
PA_GetSpriteY
Sprite, 112
PA_GifAnimSpeed
Gif, 72
PA_GifAnimStop
Gif, 72
PA_GifSetEndFrame
Gif, 73
PA_GifSetStartFrame
Gif, 72
PA_HideBg
BgTiles, 30
PA_iDeaS_DebugOutput
Debug, 47
PA_iDeaS_DebugPrintf
Debug, 48
PA_InfLargeScrollX
BgLargeMap, 21
PA_InfLargeScrollXY
BgLargeMap, 22
PA_InfLargeScrollY
BgLargeMap, 22
PA_Init16bitBg
Bitmap, 53
PA_Init16cBg
c16, 9
PA_Init8bitBg
Bitmap, 52
PA_InitBg
BgTiles, 35
PA_InitBgTrans
bgtrans, 44
PA_InitBgTransEx
bgtrans, 44
PA_InitBig8bitBg
Bitmap, 53
PA_InitComplete16c
c16, 9
PA_InitCustomKeyboard
Keyboard, 75
PA_InitCustomText
Text, 146
PA_InitFake16bitBg
Fake16bit, 63
PA_InitMode7
Mode7, 88
PA_InitParallaxX
BgTiles, 41
PA_InitParallaxY
BgTiles, 42
PA_InitSpriteDraw
Sprite, 128
PA_InitSpriteExtPrio
Sprite, 128
PA_InitTextBorders
Text, 151

PA_KeyboardIn
Keyboard, 76
PA_LargeScrollX
BgLargeMap, 22
PA_LargeScrollXY
BgLargeMap, 23
PA_LargeScrollY
BgLargeMap, 22
PA_LegacyIPCInit
General, 67
PA_Load16bitBitmap
Bitmap, 52
PA_Load8bitBgPal
Palette, 95
PA_Load8bitBitmap
Bitmap, 51
PA_LoadBackground
BgTiles, 37
PA_LoadBg
BgTiles, 32
PA_LoadBgMap
BgTiles, 36
PA_LoadBgPal
Palette, 96
PA_LoadBgPalN
Palette, 96
PA_LoadBgTiles
BgTiles, 31
PA_LoadBgTilesEx
BgTiles, 35
PA_LoadBmp
Bitmap, 58
PA_LoadBmpEx
Bitmap, 58
PA_LoadBmpToBuffer
Bitmap, 57
PA_LoadDefaultKeyboard
Keyboard, 75
PA_LoadDefaultText
Text, 147
PA_LoadFake16bitBitmap
Fake16bit, 61
PA_LoadGif
Gif, 72
PA_LoadGifXY
Gif, 72
PA_LoadJpeg
Bitmap, 57
PA_LoadKeyboard
Keyboard, 75
PA_LoadLargeBg
BgLargeMap, 20
PA_LoadLargeBgEx
BgLargeMap, 21
PA_LoadPAGfxLargeBg
BgLargeMap, 20
PA_LoadPAGfxRotBg
BgRot, 25
PA_LoadPal
Palette, 93
PA_LoadPal16
Palette, 94
PA_LoadRotBg
BgRot, 24
PA_LoadSimpleBg
BgTiles, 32
PA_LoadSprite16cPal
Palette, 94
PA_LoadSpritePal
Palette, 96
PA_LoadText
Text, 149
PA_LoadTiledBg
BgTiles, 31
PA_Locate
General, 70
PA_MicReplay
Micro, 87
PA_MicStartRecording
Micro, 87
PA_Mode7Angle
Mode7, 89
PA_Mode7Height
Mode7, 90
PA_Mode7MoveForwardBack
Mode7, 89
PA_Mode7MoveLeftRight
Mode7, 89
PA_Mode7SetPointXZ
Mode7, 90
PA_Mode7X
Mode7, 89
PA_Mode7Z
Mode7, 89
PA_modf32
Math, 86
PA_MoveSprite
Keys, 79
PA_MoveSpriteDistance
Keys, 80

PA_MoveSpriteEx
Keys, 80
PA_MoveSpritePix
Keys, 80
PA_MSG_DSLBRIGHT
General, 69
PA_MSG_INPUT
General, 69
PA_MSG_MIC
General, 69
PA_MSG_MICSTOP
General, 69
PA_MSG_PSG
General, 69
PA_mulf32
Math, 86
PA_OutputSimpleText
Text, 148
PA_OutputText
Text, 147
PA_ParallaxScrollX
BgTiles, 42
PA_ParallaxScrollXY
BgTiles, 42
PA_ParallaxScrollY
BgTiles, 42
PA_Point, 188
PA_Print
Text, 152
PA_PrintLetter
Text, 152
PA_Put16bitPixel
Bitmap, 54
PA_Put2_8bitPixels
Bitmap, 53
PA_Put4_8bitPixels
Bitmap, 54
PA_Put8bitPixel
Bitmap, 53
PA_PutDouble8bitPixels
Bitmap, 54
PA_PutFake16bitPixel
Fake16bit, 61
PA_RandMax
Math, 84
PA_RandMinMax
Math, 84
PA_RecoAddShape
Reco, 101
PA_ReLoadBgTiles
BgTiles, 36
PA_ResetBg
BgTiles, 31
PA_ResetBgSysScreen
BgTiles, 35
PA_RGB
Palette, 94
PA_ScrollKeyboardX
Keyboard, 76
PA_ScrollKeyboardXY
Keyboard, 76
PA_ScrollKeyboardY
Keyboard, 76
PA_Set16bitSpriteAlpha
Sprite, 124
PA_SetAutoCheckLid
General, 69
PA_SetBgColor
Palette, 97
PA_SetBgMosaicXY
SpecialFx, 103
PA_SetBgPalCol
Palette, 94
PA_SetBgPalNCol
Palette, 96
PA_SetBgPrio
BgTiles, 39
PA_SetBgPrioSeq
BgTiles, 39
PA_SetBgRot
BgRot, 25
PA_SetBgWrap
BgTiles, 41
PA_SetBrightness
Palette, 95
PA_SetDrawSize
Bitmap, 51
PA_SetDSLBrightness
General, 70
PA_SetKeyboardColor
Keyboard, 76
PA_SetKeyboardScreen
Keyboard, 77
PA_SetLargeMapTile
BgTiles, 38
PA_SetLedBlink
General, 69
PA_SetMapTile
BgTiles, 38
PA_SetMapTileAll

BgTiles, 33
PA_SetMapTileHflip
BgTiles, 38
PA_SetMapTilePal
BgTiles, 39
PA_SetMapTileVflip
BgTiles, 38
PA_SetOutWin
Window, 168
PA_SetPal16Neg
Palette, 95
PA_SetPalNeg
Palette, 95
PA_SetRotset
Sprite, 122
PA_SetRotsetNoAngle
Sprite, 123
PA_SetRotsetNoZoom
Sprite, 123
PA_SetScreenLight
General, 70
PA_SetScreenSpace
SpriteDual, 133
PA_SetSFXAlpha
SpecialFx, 104
PA_SetSpriteAnim
Sprite, 124
PA_SetSpriteAnimEx
Sprite, 124
PA_SetSpriteAnimFrame
Sprite, 126
PA_SetSpriteAnimSpeed
Sprite, 126
PA_SetSpriteColors
Sprite, 113
PA_SetSpriteDblsize
Sprite, 113
PA_SetSpriteGfx
Sprite, 116
PA_SetSpriteHflip
Sprite, 115
PA_SetSpriteMode
Sprite, 114
PA_SetSpriteMosaic
Sprite, 114
PA_SetSpriteMosaicXY
SpecialFx, 103
PA_SetSpriteNCycles
Sprite, 127
PA_SetSpritePal
Sprite, 112
PA_SetSpritePalCol
Palette, 97
PA_SetSpritePixel
Sprite, 127
PA_SetSpritePrio
Sprite, 116
PA_SetSpriteRotDisable
Sprite, 111
PA_SetSpriteRotEnable
Sprite, 111
PA_SetSpriteVflip
Sprite, 115
PA_SetSpriteX
Sprite, 111
PA_SetSpriteXY
Sprite, 123
PA_SetSpriteY
Sprite, 112
PA_SetTextCol
Text, 149
PA_SetTextTileCol
Text, 147
PA_SetTileLetter
Text, 146
PA_SetVideoMode
General, 69
PA_SetWin1XY
Window, 167
PA_ShowBg
BgTiles, 30
PA_ShowFont
Text, 146
PA_SimpleBoxText
Text, 151
PA_SpriteAnimPause
Sprite, 127
PA_SpriteStylusOver
Keys, 81
PA_SpriteStylusOverEx
Keys, 80
PA_SpriteTouched
Keys, 81
PA_SpriteTouchedEx
Keys, 81
PA_sqrtf32
Math, 86
PA_SRand
Math, 84
PA_StartSpriteAnim

Sprite, 125
PA_StartSpriteAnimEx
Sprite, 125
PA_StopSpriteAnim
Sprite, 125
PA_StylusInZone
Keys, 79
PA_TransferRegion, 189
PA_TrueDistance
Math, 85
PA_UpdateGfx
Sprite, 122
PA_UpdateGfxAndMem
Sprite, 122
PA_UpdateSpriteGfx
Sprite, 111
PA_UsePAGraffiti
Reco, 101
PA_WaitFor
General, 68
PA_WindowFade
Window, 169
Palette
PA_3DSetSpritePalCol, 97
PA_Load8bitBgPal, 95
PA_LoadBgPal, 96
PA_LoadBgPalN, 96
PA_LoadPal, 93
PA_LoadPal16, 94
PA_LoadSprite16cPal, 94
PA_LoadSpritePal, 96
PA_RGB, 94
PA_SetBgColor, 97
PA_SetBgPalCol, 94
PA_SetBgPalNCol, 96
PA_SetBrightness, 95
PA_SetPal16Neg, 95
PA_SetPalNeg, 95
PA_SetSpritePalCol, 97
Palette system, 92
Palette system for Dual Screen, 98
PaletteDual
PA_DualLoadBgPal, 100
PA_DualLoadPal, 98
PA_DualLoadPal16, 99
PA_DualLoadSpritePal, 99
PA_DualSetBgColor, 100
PA_DualSetPal16Neg, 99
PA_DualSetPalNeg, 99
Reco
PA_RecoAddShape, 101
PA_UsePAGraffiti, 101
Rotating Backgrounds, 24

Shape Recognition, 101
SNDCMD_PLAY
ASlib, 174
SNDCMD_SETMASTERVOLUME
ASlib, 174
SNDCMD_SETPAN
ASlib, 174
SNDCMD_SETRATE
ASlib, 174
SNDCMD_SETVOLUME
ASlib, 174
SNDCMD_STOP
ASlib, 174
SoundCommand
ASlib, 174
SoundInfo, 191
Special controllers, 82
Special Effects, 102
SpecialFx
PA_DisableBgMosaic, 103
PA_DisableSpecialFx, 104
PA_EnableBgMosaic, 102
PA_EnableSpecialFx, 103
PA_SetBgMosaicXY, 103
PA_SetSFXAlpha, 104
PA_SetSpriteMosaicXY, 103
Sprite
PA_CloneSprite, 117
PA_Create16bitSprite, 120
PA_Create16bitSpriteEx, 119
PA_Create16bitSpriteFromGfx, 120
PA_CreateGfx, 118
PA_CreateSprite, 118
PA_CreateSpriteEx, 118
PA_CreateSpriteExFromGfx, 121
PA_CreateSpriteFromGfx, 120
PA_DeleteGfx, 122
PA_DeleteSprite, 122
PA_GetSprite16cPixel, 128
PA_GetSpriteAnimFrame, 126
PA_GetSpriteAnimSpeed, 126
PA_GetSpriteColors, 113
PA_GetSpriteDblsize, 113
PA_GetSpriteGfx, 116
PA_GetSpriteHflip, 115

PA_GetSpriteLx, 117
PA_GetSpriteLy, 117
PA_GetSpriteMode, 114
PA_GetSpriteMosaic, 114
PA_GetSpriteNCycles, 127
PA_GetSpritePal, 112
PA_GetSpritePixel, 128
PA_GetSpritePrio, 116
PA_GetSpriteVflip, 115
PA_GetSpriteX, 112
PA_GetSpriteY, 112
PA_InitSpriteDraw, 128
PA_InitSpriteExtPrio, 128
PA_Set16bitSpriteAlpha, 124
PA_SetRotset, 122
PA_SetRotsetNoAngle, 123
PA_SetRotsetNoZoom, 123
PA_SetSpriteAnim, 124
PA_SetSpriteAnimEx, 124
PA_SetSpriteAnimFrame, 126
PA_SetSpriteAnimSpeed, 126
PA_SetSpriteColors, 113
PA_SetSpriteDblsize, 113
PA_SetSpriteGfx, 116
PA_SetSpriteHflip, 115
PA_SetSpriteMode, 114
PA_SetSpriteMosaic, 114
PA_SetSpriteNCycles, 127
PA_SetSpritePal, 112
PA_SetSpritePixel, 127
PA_SetSpritePrio, 116
PA_SetSpriteRotDisable, 111
PA_SetSpriteRotEnable, 111
PA_SetSpriteVflip, 115
PA_SetSpriteX, 111
PA_SetSpriteXY, 123
PA_SetSpriteY, 112
PA_SpriteAnimPause, 127
PA_StartSpriteAnim, 125
PA_StartSpriteAnimEx, 125
PA_StopSpriteAnim, 125
PA_UpdateGfx, 122
PA_UpdateGfxAndMem, 122
PA_UpdateSpriteGfx, 111
Sprite system, 105
Sprite system for Dual Screen, 130
SpriteDual
PA_DualCloneSprite, 141
PA_DualCreate16bitSprite, 135
PA_DualCreate16bitSpriteEx, 135
PA_DualCreateSprite, 133
PA_DualCreateSpriteEx, 134
PA_DualCreateSpriteExFromGfx, 136
PA_DualCreateSpriteFromGfx, 136
PA_DualDeleteSprite, 137
PA_DualGetSpriteAnimFrame, 143
PA_DualGetSpriteAnimSpeed, 143
PA_DualSetRotset, 138
PA_DualSetRotsetNoAngle, 138
PA_DualSetRotsetNoZoom, 138
PA_DualSetSpriteAnim, 141
PA_DualSetSpriteAnimEx, 141
PA_DualSetSpriteAnimFrame, 142
PA_DualSetSpriteAnimSpeed, 143
PA_DualSetSpriteColors, 139
PA_DualSetSpriteDblsize, 139
PA_DualSetSpriteGfx, 140
PA_DualSetSpriteHflip, 140
PA_DualSetSpriteMode, 139
PA_DualSetSpriteMosaic, 140
PA_DualSetSpritePal, 139
PA_DualSetSpritePrio, 140
PA_DualSetSpriteRotDisable, 138
PA_DualSetSpriteRotEnable, 137
PA_DualSetSpriteVflip, 140
PA_DualSetSpriteX, 133
PA_DualSetSpriteXY, 133
PA_DualSetSpriteY, 133
PA_DualSpriteAnimPause, 143
PA_DualStartSpriteAnim, 142
PA_DualStartSpriteAnimEx, 141
PA_DualStopSpriteAnim, 142
PA_DualUpdateGfx, 137
PA_DualUpdateSpriteGfx, 137
PA_SetScreenSpace, 133
Text
PA_8bitCustomFont, 146
PA_8bitText, 149
PA_AddBitmapFont, 150
PA_BoxText, 148
PA_BoxTextNoWrap, 149
PA_CenterSmartText, 150
PA_ClearTextBg, 151
PA_EraseTextBox, 151
PA_InitCustomText, 146
PA_InitTextBorders, 151
PA_LoadDefaultText, 147
PA_LoadText, 149

PA_OutputSimpleText, 148
PA_OutputText, 147
PA_Print, 152
PA_PrintLetter, 152
PA_SetTextCol, 149
PA_SetTextTileCol, 147
PA_SetTileLetter, 146
PA_ShowFont, 146
PA_SimpleBoxText, 151
Text output system, 144
Tiled Background Modes, 27
TileDual
PA_DualBGScrollX, 160
PA_DualBGScrollXY, 161
PA_DualBGScrollY, 161
PA_DualDeleteBg, 160
PA_DualEasyBgLoad, 159
PA_DualEasyBgScrollX, 161
PA_DualEasyBgScrollXY, 162
PA_DualEasyBgScrollY, 161
PA_DualHideBg, 160
PA_DualInfLargeScrollX, 162
PA_DualInfLargeScrollXY, 162
PA_DualInfLargeScrollY, 162
PA_DualInitParallaxX, 163
PA_DualInitParallaxY, 164
PA_DualLargeScrollX, 163
PA_DualLargeScrollXY, 163
PA_DualLargeScrollY, 163
PA_DualLoadBackground, 162
PA_DualLoadBg, 157
PA_DualLoadLargeBg, 158
PA_DualLoadLargeBgEx, 159
PA_DualLoadPAGfxLargeBg, 158
PA_DualLoadRotBg, 156
PA_DualLoadSimpleBg, 156
PA_DualLoadTiledBg, 155
PA_DualParallaxScrollX, 164
PA_DualParallaxScrollXY, 165
PA_DualParallaxScrollY, 164
PA_DualSetBgPrio, 165
PA_DualShowBg, 160

Window
PA_DisableWin0, 167
PA_DisableWin1, 168
PA_DisableWinObj, 168
PA_EnableWin0, 167
PA_EnableWin1, 167
PA_EnableWinObj, 169
PA_SetOutWin, 168
PA_SetWin1XY, 167
PA_WindowFade, 169
Window system, 166